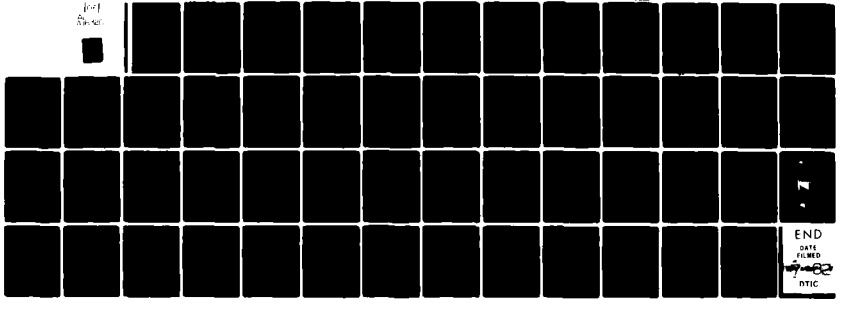
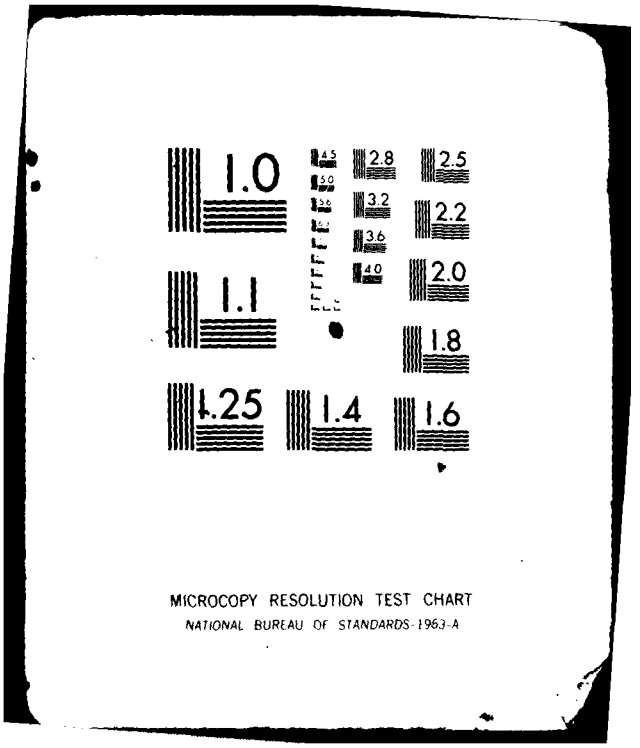
AD-A116 320 AERONAUTICAL RESEARCH LABS MELBOURNE (AUSTRALIA) F/G 9/2
A SCAN LINE ALGORITHM FOR COMPUTER GENERATED FLIGHT VISUALS,(U)
FEB 81 J SANDOR, P G JOST

UNCLASSIFIED ARL-SYS-NOTE-77 NL



END
DATE FILMED
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ARL-SYS-NOTE-77

AR-002-259

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

AERONAUTICAL RESEARCH LABORATORIES

MELBOURNE, VICTORIA

SYSTEMS NOTE 77

A SCAN LINE ALGORITHM FOR COMPUTER GENERATED FLIGHT VISUALS

by

JOHN SANDOR and PETER G. JOST

COPY No

FEBRUARY, 1981

AR-002-259

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
AERONAUTICAL RESEARCH LABORATORIES

SYSTEMS NOTE 77

A SCAN LINE ALGORITHM FOR COMPUTER GENERATED FLIGHT VISUALS

by

JOHN SANDOR and PETER G. JOST

SUMMARY

A scan line algorithm is presented for the computer synthesis of flight simulation visuals. The algorithm is described in the context of the total image generation process which includes data base structure, viewing geometry and polygon clipping. Within this framework the performance of a software realization of the algorithm is analysed and the relative computational loads of different algorithm functions are assessed. Finally the extension of the algorithm to accommodate shading and edge smoothing is described.

POSTAL ADDRESS: Chief Superintendent, Aeronautical Research Laboratories,
Box 4331, P.O., Melbourne, Victoria, 3001, Australia.

DOCUMENT CONTROL DATA SHEET

Security classification of this page: Unclassified

1. Document Numbers
   (a) AR Number:
   AR-002-259
   (b) Document Series and Number:
   Systems Note 77
   (c) Report Number:
   ARL-Sys-Note-77

2. Security Classification
   (a) Complete document:
   Unclassified
   (b) Title in isolation:
   Unclassified
   (c) Summary in isolation:
   Unclassified

3. Title: A SCAN LINE ALGORITHM FOR COMPUTER GENERATED FLIGHT VISUALS

4. Personal Author(s):
   J. Sandor
   P. G. Jost

5. Document Date:
   February, 1981

6. Type of Report and Period Covered:

7. Corporate Author(s):
   Aeronautical Research Laboratories

8. Reference Numbers
   (a) Task:
   DST 80/099
   (b) Sponsoring Agency:
   DSTO

9. Cost Code:
   71 6490

10. Imprint
    Aeronautical Research Laboratories,
    Melbourne

11. Computer Program(s)
    (Title(s) and language(s)):
    —

12. Release Limitations (of the document)
    Approved for Public Release

12.0. Overseas:	N.O.		P.R.	1	A		B		C		D		E	

13. Announcement Limitations (of the information on this page):
    No Limitations

14. Descriptors:

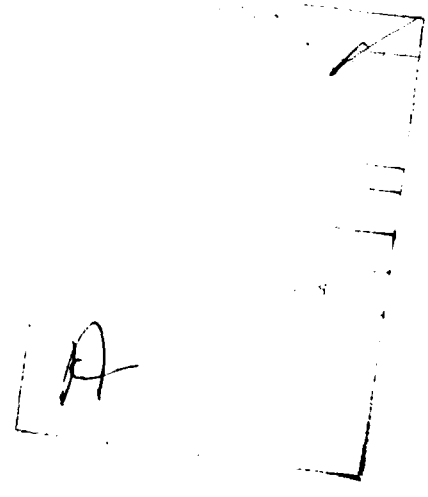
Computer graphics	Flight simulation	
Graphic methods	Image processing	
Computer generated images	Display systems	
Algorithms		

15. Cosati Codes:
    0902
    1201

16. *ABSTRACT*

A scan line algorithm is presented for the computer synthesis of flight simulation visuals. The algorithm is described in the context of the total image generation process which includes data base structure, viewing geometry and polygon clipping. Within this framework the performance of a software realization of the algorithm is analysed and the relative computational loads of different algorithm functions are assessed. Finally the extension of the algorithm to accommodate shading and edge smoothing is described.

CONTENTS



NOTATION

$O_L X_L Y_L Z_L$	local coordinate frame of an object
$O_E X_E Y_E Z_E$	environment coordinate frame
$OXYZ$	observer coordinate frame
$O\hat{X}\hat{Y}\hat{Z}$	coordinate frame with origin coincident with O and parallel to $O_E X_E Y_E Z_E$
$O_s X_s Y_s$	screen coordinates
$n(i)$	number of vertices defining i-th object
f_i	number of faces comprising i-th object
$g(i)$	number of vertices defining the i-th face
O_{im}	perspective image of O on display screen
$(\tilde{x}, \tilde{y})$	coordinates of O_{im} in $O_s X_s Y_s$
v_i	i-th vertex
$\mathbf{V}_i$	vector from coordinate frame origin to vertex v_i
c	face centroid
(c_x, c_y, c_z)	coordinates of face centroid
$\mathbf{N}$	outward surface normal vector
d	distance between observer and display screen
s_i	colour at i-th vertex of a face
$v_i v_{i+1}$	directed line segment (edge) from vertex v_i to v_{i+1}
Δy	scan line dimension in screen units
Δx	pixel dimension in screen units
(i, j)	coordinates of a screen point in pixel, scan line units
$c(i, j)$	colour at pixel (i, j)
Δc	colour increment along edge
δc	colour increment/pixel along a scan line segment spanning a surface
t	parameter for equation of a plane in $OXYZ$
$\|\mathbf{V}\|$	the Euclidean norm of vector $\mathbf{V}$
$\mathbf{V}^T$	transpose of vector V.

1. INTRODUCTION

The effectiveness of complex military aircraft in stringent operational environments is dependent on the efficient integration of aircrew, avionics and displays, weapon systems and flight vehicle. Manned flight simulation has become a major research tool in this work in the U.S., U.K. and other NATO countries. At A.R.L. flight simulation is directed towards providing a versatile facility for research in man-machine aircraft systems and to provide support to Service uses of simulators in training.

Central to flight simulation is the use of computer generated imagery (CGI) for the presentation of the external visual environment. The versatility and flexibility of this approach makes it particularly well suited to research orientated simulation. The present raster-based CGI system at A.R.L. is integrated with a fixed base multicrew simulator and provides a wide angle, full colour, image of limited scene content. This report describes the non-real time software realization of a scan line algorithm which is to form the basis of a raster-based CGI system with enhanced capability in terms of image content, image update rate and image quality.

The essential function of the algorithm is to solve the hidden surface problem given a polygon description of the environment. The algorithm first sorts the edges of the visible polygons from top to bottom. Then for each scan line the edge intercepts are sorted from left to right and finally a depth comparison determines the visible polygon segments along the scan. The key features of the algorithm are the utilization of image coherence for efficient sorting and the storage of depth-sorted polygons in a priority list.

The algorithm is described in the context of the total CGI process which includes database structure, viewing geometry and clipping. Within this framework the performance of a software realization of the algorithm is analysed and the relative computational loads of different algorithm functions are assessed. Finally it is explicitly shown that the image enhancement techniques of shading and edge smoothing can be readily accommodated within the structure of the scan line algorithm.

2. VISUAL ENVIRONMENT REPRESENTATION

2.1 *Object Definition*

The environment has the following visual features:

(i) a planar surface representing a local flat earth

(ii) 2-dimensional (surface) objects such as runways and markings

(iii) 3-dimensional objects representing man-made features and terrain on the earth's surface, and cloud and aircraft above the earth.

This categorization of environment features constitutes a hierarchy in terms of visual occultation; namely that a class (iii) object can occlude from view other objects of class (iii), and class (ii) objects, whilst class (ii) objects many only occlude other class (ii) objects and the earth plane. The implication of this hierarchical structure is that in situations involving objects of different classes, the hidden surface problem can be readily resolved by reference to the above precedence relationships. Moreover, in this context, the structure can be expanded to encompass precedence relationships amongst objects of the same class. For instance in the case of a marked runway, the runway markings would have precedence over the runway since the latter can never occult the former from view.

In addition to the above objects, the environment model accommodates two additional features: firstly, the influence of a diffuse atmosphere, and secondly a horizon line. The former

is visually characterized by the desaturation of surface colours of objects at large distances from the observer, whilst the latter represents the straight line between the earth plane and the sky in the absence of terrain or man-made cultural features. Since topographical features are modelled independently of the earth plane, the precedence relationships ensure that the horizon silhouette as viewed by the observer is faithfully reproduced. The synthesis of the horizon is described in Section 3.7 whilst atmospheric desaturation effects are described in Reference 1.

In the context of CGI, the numerical characterization or modelling of an object is dependent on the particular algorithm(s) employed and the image data format required by the display. For the present case, time critical computation constraints together with a format compatible with raster displays, dictates that objects be modelled as convex polyhedra. Consequently it will be seen in the sequel that at each stage of the CGI process only linear equations arise. These can be solved by rational operations. Contrasting this, the solution of equations arising from the non-linear description of a curved surfaced object will require irrational operations, which are computationally expensive.

Adopting this polyhedral framework for object representation, each object model is described by the locations of its vertices, sequence of vertices comprising each face and the colour of each face. Each vertex of an object is located within a "local" coordinate frame $O_L X_L Y_L Z_L$, with origin O_L being one of the object's vertices. Then the location and extent of an object in the environment coordinate frame $O_E X_E Y_E Z_E$ is deduced from the location of O_L and relative orientation of $O_L X_L Y_L Z_L$ with respect to the environment coordinate frame. Since the description of an object in $O_L X_L Y_L Z_L$ is invariant of $O_E X_E Y_E Z_E$ frame, the model of each object is "portable" from one synthesized environment to another.

For a scene described in $O_E X_E Y_E Z_E$, many images can be generated depending on the position from which the scene is viewed. In any specific case the location and orientation of the observer centred reference frame $OXYZ$ is required to compute the visible scene.

The geometry of the overall scene description is depicted in Figure 2.1, whilst a detailed exposition of the transformations required to compute the coordinates of an object relative to $OXYZ$ given its coordinates in $O_L X_L Y_L Z_L$ is left to Appendix I. However it is worthy to note here that for objects which remain stationary with respect to the environment coordinate frame, transformations from local to environment coordinate frame need be carried out only during the preparation of the data base.

2.2 Data Structure

It is clear from the preceding section that the data required to describe a scene object, in the sense of spatial extent, are the coordinates of the object's vertices, together with the topology of connections between vertices. However, as alluded to in Reference 1, there are image synthesis algorithms (such as clipping) where the quantities of interest are not vertices *per se* but edges and polygons. This implies that the environment description be structured such that each algorithm of the CGI process can deduce, as required, object vertices, edges and faces. The method adopted here and described below, accomplishes this by defining for each environment object an indexed list of vertex coordinates together with sequences of indices for face and edge definition.

For an object with n vertices, indexed $0, 1, \ldots, n-1$, the coordinates of the vertices are stored in three parallel lists denoted here by x, y, z, and it is assumed that $x(0), y(0), z(0)$ are the coordinates of the local origin O_L relative to the environment coordinate frame. To determine the faces of the object additional information regarding the topology of the connections between vertices and faces has to be included in the data base. To accommodate this let

(i) f be the number of faces defining the object

(ii) g be a list such that $g(i)$ specifies the number of vertices describing the i-th face

(iii) m be a sequence of vertex indices, ordered such that

$m(1), \ldots, m(g(1))$ are the indices specifying the vertices of face (1)

$m(g(1)+1), \ldots, m(g(1)+g(2))$ are the indices specifying the vertices of face (2)

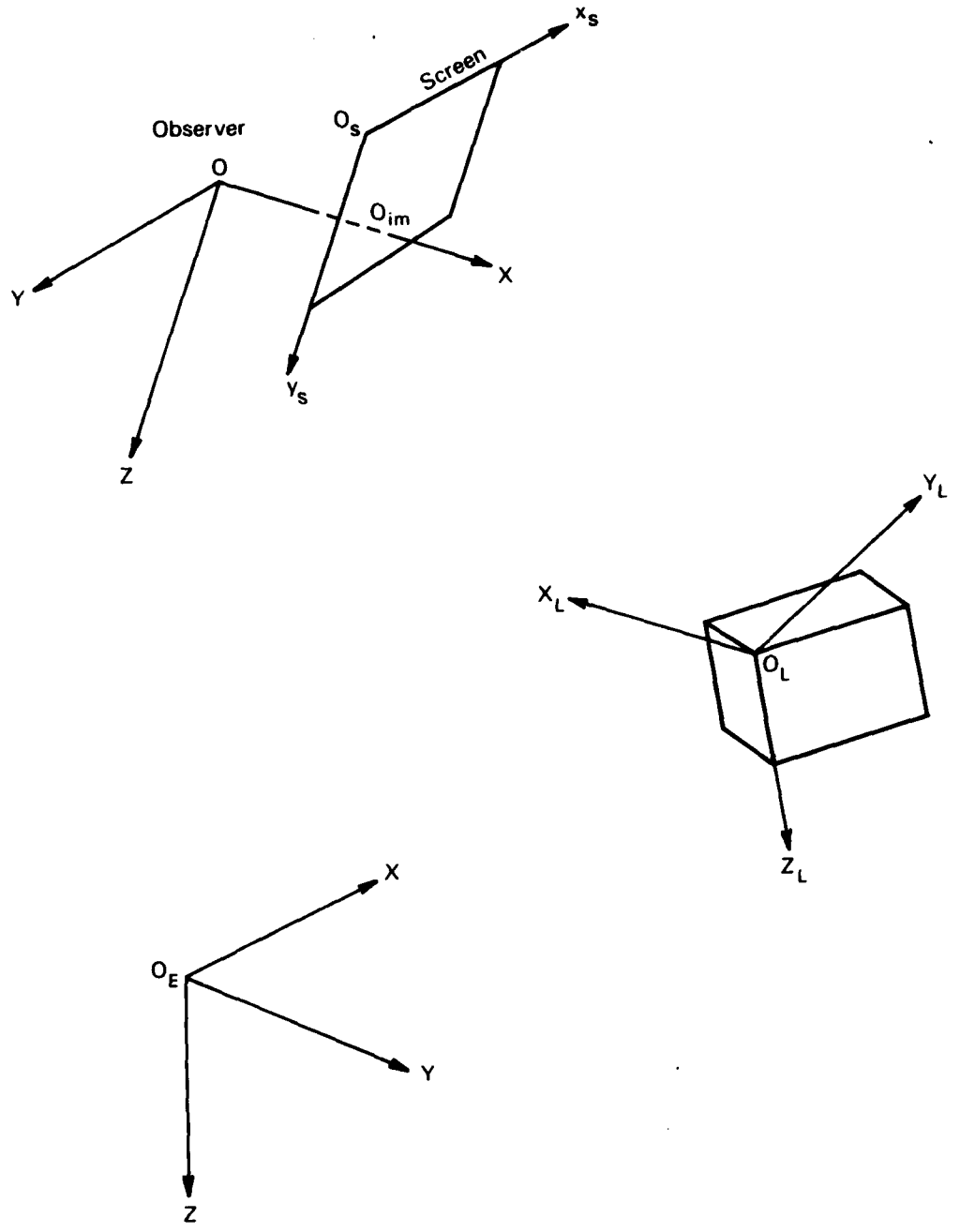


FIG. 2.1 GEOMETRY OF VIEWING THE ENVIRONMENT.

and so on. Clearly the elements of m function as pointers to the object vertex lists x, y, z.

The linked data structure encompassing the above quantities is illustrated in Figure 2.2.

In addition to the above properties of m, the sequence $\{m(k), \ldots, m(k + g(j))\}$, $k = 1 + \sum_{l=1}^{j-1} g(l)$, corresponding to the indices defining the vertices of the j-th face is assumed to be such that traversing the face from $m(k)$-th vertex to the $m(k + g(j))$-th vertex, the interior of the face is always to the left. For the object illustrated in Figure 2.3, the sequences

$$\{2, 3, 6, 1, 0\}$$

and

$$\{6, 1, 0, 2, 3\}$$

are valid in the sense specified above, whilst the sequence

$$\{1, 6, 3, 2, 0\}$$

is not valid.

Further, by consistently applying this ordering to each face it will be shown in Chapter 3 that each visible edge in the perspective image forms one of two readily deduced relationships with respect to the scan direction.

Although no explicit mention has been made of features such as surface colour or transparency, it is implicit that the relevant data can be stored in a list similar to g. Further the angles defining the orientation of the local coordinate frame of each object with respect the environment frame are stored analogously to n and f, denoted as object attributes in Figure 2.2.

For convenience in program implementation, the lists x, y, z, g and m for all the scene objects are formed into contiguous lists with pointers stored with the object attributes delineating one object description from another, as shown in Figure 2.4.

3. EDGE GENERATION

In the last Chapter the visual environment data structure was defined and the method of processing the data to the point of delineating polygon faces from object description was described. Edge generation comprises five distinct phases and results in the subdivision of each face into a set of visible edges which are processed by the scan line algorithm. The importance of these edges can be seen by considering the relationship between scene edges and display scan lines as illustrated in Figure 3.1. It is readily observed that polygon edges divide each scan line into homogeneous regions within which only one face is visible. The location of the region boundaries, together with the surface shade on either side of each boundary, are of ultimate interest to the scan line algorithm.

The five phases of edge generation are treated below, and for convenience it will be assumed that the entity of interest is the j-th face of the i-th object which is described, in observer coordinates by the $p + 1$ vertices $v_0(x_0, y_0, z_0), \ldots, v_p(x_p, y_p, z_p)$.

3.1 Backface Elimination

From the point of view of CGI, object faces fall into one of two categories:

(i) "in-view" faces which are visible to the observer

(ii) faces which are occulted from view by the object's own volume.

The latter are termed "backfaces" and have to be eliminated as they do not contribute to the image. Moreover it is clear that this has to be performed each time a new image is to be generated as the categorization is dependent on the viewpoint location.

To determine whether a face is a backface it is necessary to compute the outward surface normal and the face centroid. Let $c(c_x, c_y, c_z)$ and N denote the centroid and face normal, respectively, and consider the face depicted in Figure 3.2. The face centroid is determined by

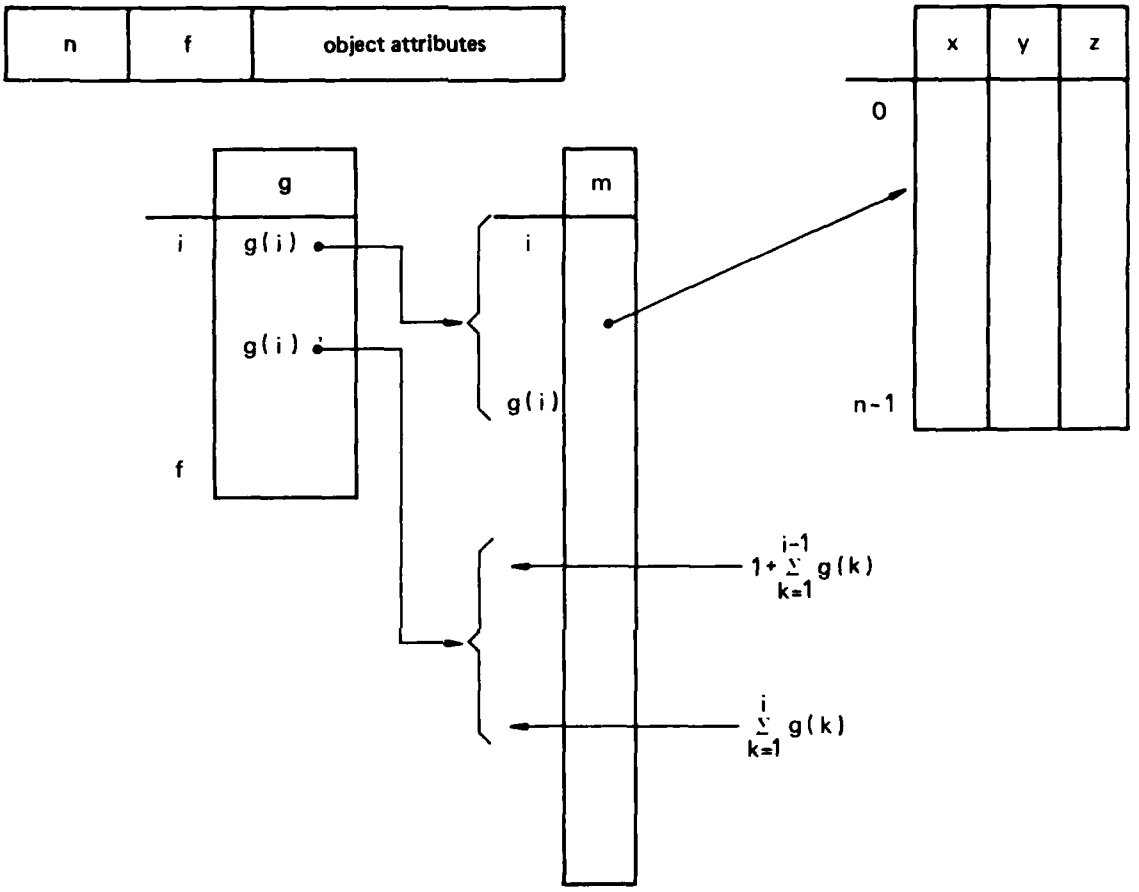


FIG. 2.2 LINKED DATABASE STRUCTURE FOR AN OBJECT.

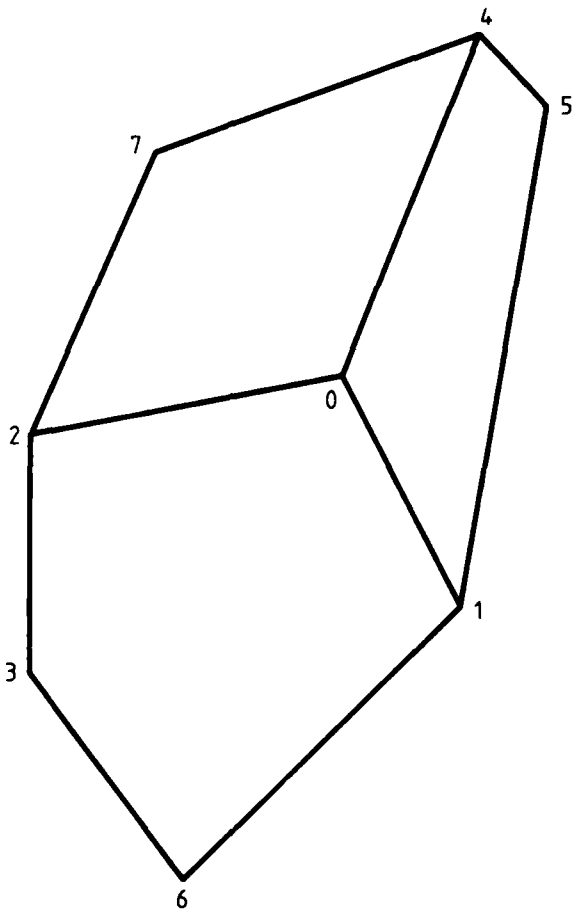


FIG. 2.3 EXAMPLE OF INDEXING OF VERTICES OF AN OBJECT.

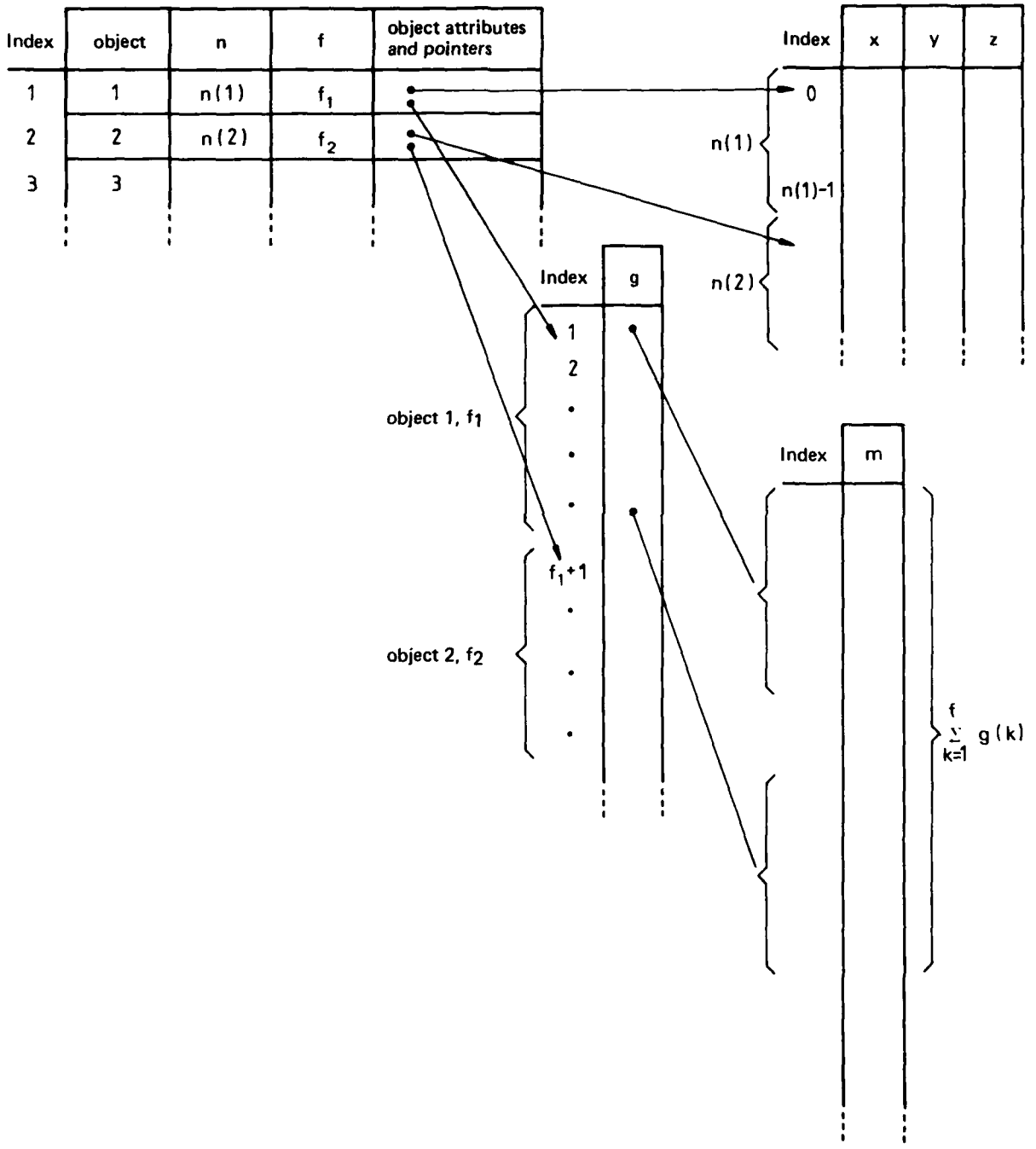


FIG. 2.4 SCENE DESCRIPTION DATA STRUCTURE

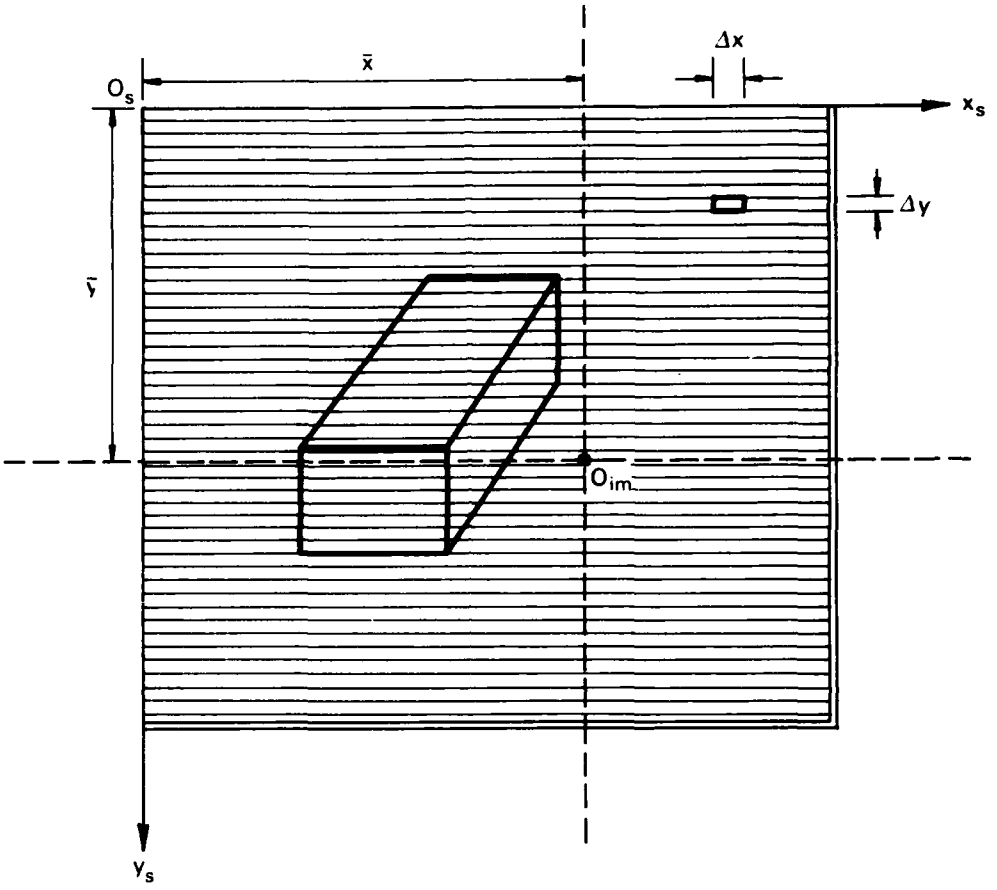


FIG. 3.1 DEFINITION OF SCREEN COORDINATES AND RELATIONSHIP BETWEEN IMAGE POLYGONS, THE ORIGIN OF THE OBSERVER COORDINATE FRAME, SCAN LINES AND PIXELS. (O_{im} IS THE PERSPECTIVE IMAGE OF THE ORIGIN OF THE OBSERVER COORDINATE FRAME.)

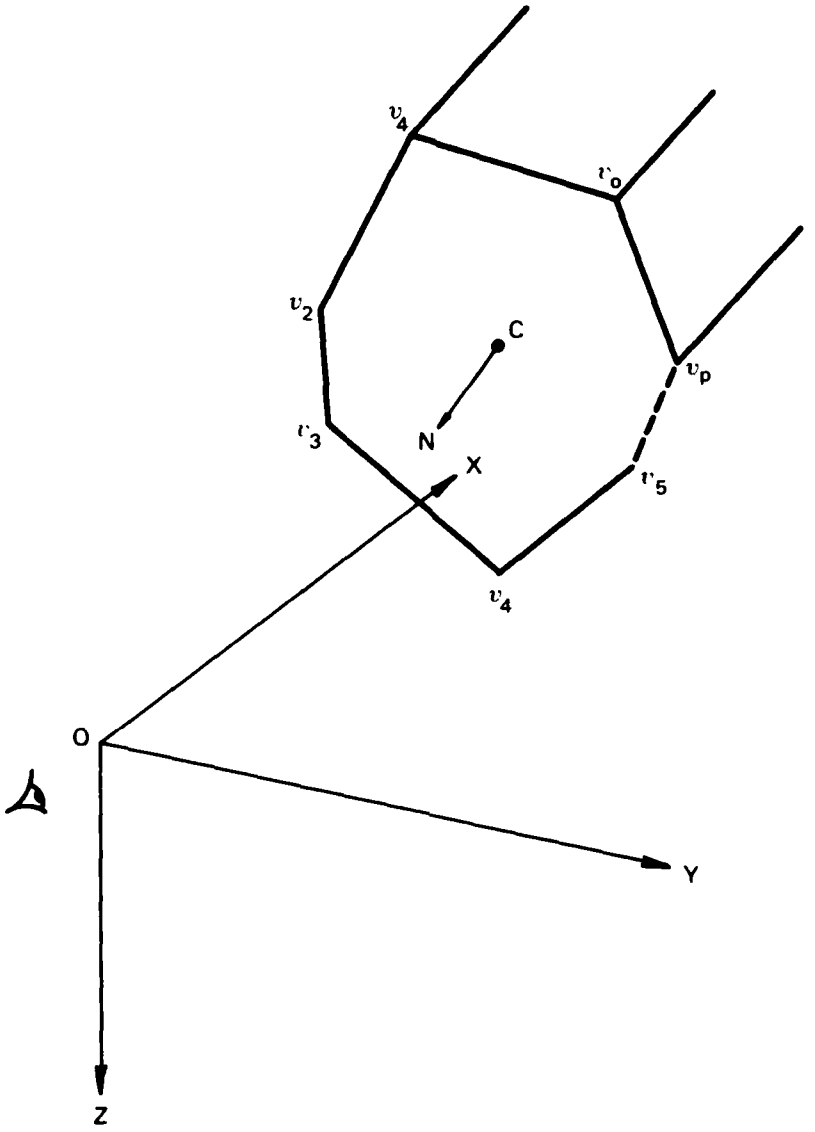


FIG. 3.2 CENTROID AND OUTWARD SURFACE NORMAL.

$$c(c_x, c_y, c_z) = \frac{1}{p+1} \sum_{i=0}^{p} v_i(x_i, y_i, z_i) \qquad 3.1.1$$

and the outward normal direction of the face is given by

$$\mathbf{N}_d = (v_i - v_{i-1}) \times (v_{i+1} - v_i) : 1 \leqslant i \leqslant p-1$$

Normalizing $\mathbf{N}_d$ one obtains the outward surface normal.

$$\mathbf{N} = \frac{\mathbf{N}_d}{\mathbf{N}_d} \qquad 3.1.2$$

Denote by $\mathbf{C}$ the vector from the origin of the observer coordinate frame to the face centroid, defining the direction of the "line of sight" to the face and let Φ be the angle between the line of sight and the face normal direction. The angle is defined by

$$\cos\Phi = \frac{\mathbf{C} \cdot \mathbf{N}}{\mathbf{C}} \qquad 3.1.3$$

and it follows that if

$$\cos\Phi \geqslant 0$$

then the face is a backface and need not be considered in further processing. In contrast, for those faces which are "in-view", the distance of the centroid to the observer, $\mathbf{C}$, is convenient for the purposes of depth comparison, whilst in conjunction with $\mathbf{N}$ it is incorporated in some useful shading functions [2].

3.2 Clipping

Clipping encompasses the processing of each object in the environment such that objects which (i) are behind the observer, (ii) are at a distance which places them beyond the horizon, and (iii) do not project within the boundaries of the display screen, are culled, whilst objects that only partially project onto the screen are truncated at the screen boundaries. In three dimensions the visible extent of the environment is defined to be that region which falls within a pyramid, termed "the viewbox", with apex at the observer's eye and enclosing the display screen as depicted in Figure 3.3. Employing the terminology of [3], the plane containing the display and the vertical plane at the horizon will be referred to as the "hither plane" and the "yonder plane", respectively. These two boundaries limit the extent, in the x-direction, of the visual environment. It should, however, be noted that in view of the local flat earth assumption yonder plane clipping is only applicable in situations, where it is known a priori, that objects at a greater distance from the observer than the horizon subtend on the display screen an area less than one pixel in size. This will not be true in the case of large cultural features such as mountains and hence yonder plane clipping is not valid.

In the present CGI process the culling of out-of-view objects or portions of objects is achieved by the re-entrant polygon clipping algorithm of Reference 3. This approach involves determining the visibility of each object face with respect to each viewbox boundary plane. To illustrate the method consider the face defined by the sequence of vertices $v_0, \ldots, v_p$ and the viewbox boundary plane

$$ax + by + cz + d = 0$$

where it is assumed that the signs of the coefficients a, b, c and d are such that a point $r(x_r, y_r, z_r)$ is on the visible side of the plane if

$$ax_r + by_r + cz_r + d \geqslant 0 \qquad 3.2.1$$

Now the line coincident with the edge $v_i v_{i-1}$ can be written in concise parametric form, with parameter t, as

$$\mathbf{q} = \mathbf{V}_i + t(\mathbf{V}_{i-1} - \mathbf{V}_i); \qquad \mathbf{q}^T = (x, y, z) \qquad 3.2.2$$

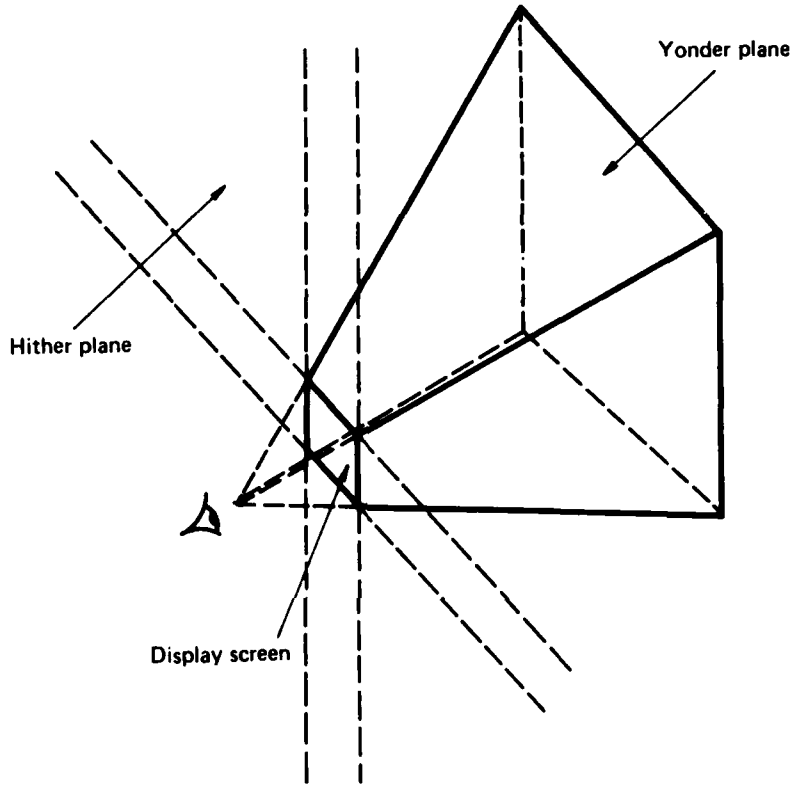


FIG. 3.3 VIEWBOX DEFINITION.

and hence the intersection of the line containing this edge and the boundary plane is given by (3.2.2), with

$$t = \frac{d + \mathbf{a}^T \cdot \mathbf{V}_i}{\mathbf{a}^T \cdot (\mathbf{V}_{i-1} - \mathbf{V}_i)} \quad ; \quad \mathbf{a}^T = (a, b, c) \qquad 3.2.3$$

and the edge intersects the plane if

$$0 \leq t \leq 1$$

Moreover if

(i) $\mathbf{a}^T \cdot (\mathbf{V}_{i-1} - \mathbf{V}_i) = 0$ then the edge is parallel to the boundary plane, and

(ii) if in addition $d + \mathbf{a}^T \cdot \mathbf{V}_i = 0$ then the edge lies on the boundary plane.

Note that in the case of curved surface approximation edges have vertex shades assigned, and the parameter t of (3.2.3) is required to compute the shade at the boundary plane crossings. That is, if s_i, s_{i-1} are the shades assigned to vertices v_i, v_{i-1} respectively, then the shade of the intersect point, q, is

$$s_q = s_i + t(s_{i-1} - s_i)$$

The clipping algorithm proceeds in the following way for edge $v_i v_{i-1}$:

step 1: compute $\mathbf{a}^T \cdot (\mathbf{V}_{i-1} - \mathbf{V}_i)$ and if this is zero then proceed to step 4.

step 2: determine t via (3.2.3) and if $t \notin (0, 1)$ proceed to step 4.

step 3: the valid intercept point, q, is given by (3.2.2) and is placed in an output file.

step 4: determine the visibility of v_{i-1} via the inequality (3.2.1).

step 5: if v_{i-1} is visible then place v_{i-1} in the output file.

step 6: if $i + 1 < p$, then increment i by 1, and return to step 1.

step 7: for $i + 1 = p$, repeat steps 1 to 6 for the edge $v_p v_0$, and stop.

The output file will then contain a sequence of vertices $r_0, \ldots, r_k$ defining the $k + 1$ vertices of the closed polygon formed by clipping the face $v_0, \ldots, v_p$ against the boundary plane. The process is repeated for each boundary plane in turn until only the visible extent of the face remains.

As a simple illustration of the process consider the two dimensional example of Figure 3.4.

3.3 Depth Priority

The visible face at any picture element (pixel) of the display is chosen, by the scan line algorithm, from the totality of in-view faces projecting on the pixel, on the basis of closest proximity to the observer. In the present approach proximity is quantified by the distance of the face centroid (equation 3.1.1) to the observer. Although this may not result in unique values for each face it is observed that in the majority of cases where the centroid-observer distances are equal, the associated faces are located in the environment such that they do not project onto the same region of the display. Moreover in cases involving 2- and 3-dimensional objects (such as a building and a runway) the precedence relationships, discussed in section 2.1, will resolve the question of proximity.

Each centroid is tagged with the index of the face to which it belongs, the tags are ordered into an indexed list such that the tags form pointers to the corresponding faces. The index of this sorted list will be referred to as the face "priority". Figure 3.5 illustrates the linkage of object to the PRIORITY LIST, such that associated with each face is a priority and with each priority a face.

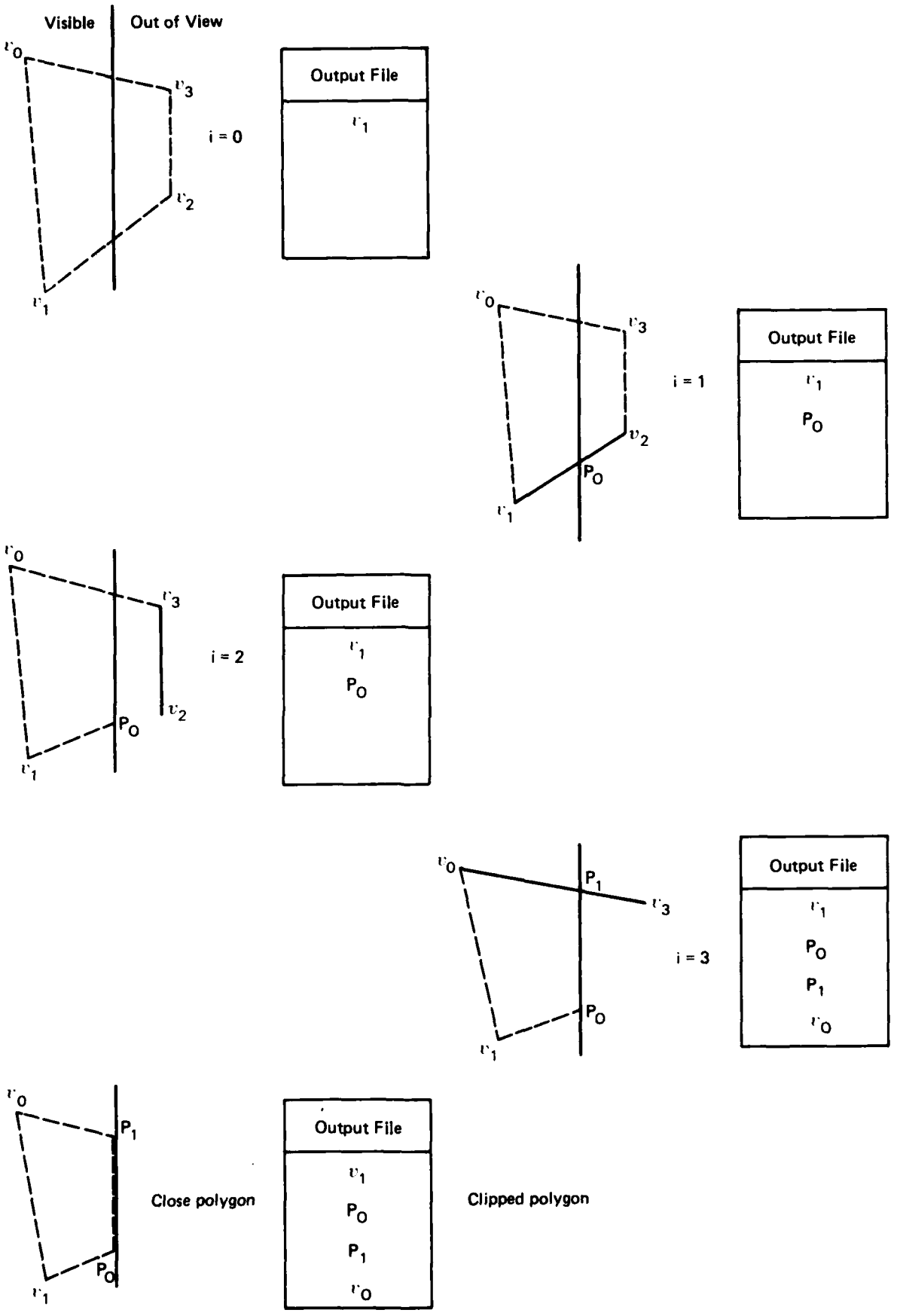


FIG. 3.4 POLYGON CLIPPING

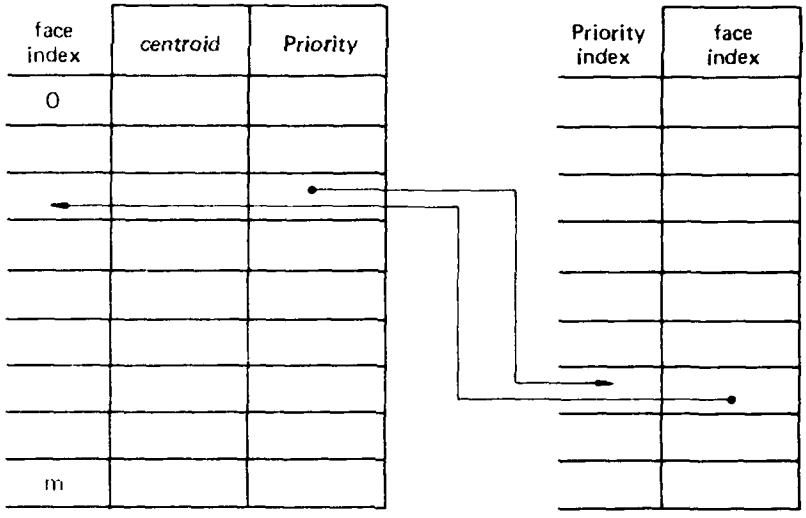


FIG. 3.5 FACE – PRIORITY LINK

3.4 Perspective Transformation

The display of the visual environment requires that the three dimensional clipped scene be transformed into a two-dimensional image such that features are numerically characterized in terms of scan line and pixel number. For the situation depicted in Figure 3.1, with the display screen perpendicular to the OZ axis, the two-dimensional image, $p_s(x_s, y_s)$, of a point $p(x, y, z)$ is given by the perspective transformation

$$x_s = \frac{x}{z} d + \bar{x}$$
$$y_s = \frac{y}{z} d + \bar{y} \qquad (3.4.1)$$

where d is the perpendicular distance between observer and screen, and $(\bar{x}, \bar{y})$ are the screen coordinates of the perspective image of the origin of the observer coordinate frame as depicted in Figure 3.1. If the dimensions of each pixel in display coordinate units are Δx and Δy, then the image of p is located at

i = $x_s / \Delta x$ pixels from the left edge of the display

j = $y_s / \Delta y$ scan lines from the top of the display.

Applying the above transformations to each vertex in the clipped scene results in the required two-dimensional image description.

3.5 Edge Orientation

Recall from Section 2.1, and Section 3.1 that for the vertex sequence $\{v_1, \ldots, v_n\}$ defining an object face, the outward surface normal direction is given by the cross product

$$(\mathbf{V}_i - \mathbf{V}_{i-1}) \times (\mathbf{V}_{i+1} - \mathbf{V}_i) .$$

Except for backfaces this corresponds to the normal vector "pointing" in the direction of the observer and hence in the perspective image the vertex sequence specifies counter clockwise travel around the polygon. Hence, for convex polygons, each edge in the perspective image has one of two orientations with respect to the left-to-right scan direction on each scan line:

(i) the scan "enters" the polygon face at the edge

(ii) the scan "leaves" (or "exits") the polygon face at the edge

for a conventional raster display.

The counter clockwise definition of each image polygon implies that for the edge $v_K(i_K, j_K)$, $v_{K+1}(i_{K+1}, j_{K+1})$ (see Fig. 3.6), the scan "enters" the polygon at the edge if

$$(j_{K+1} - j_K) > 0$$

else it "leaves" the polygon at the edge. Classifying each edge in this way will permit the scan line algorithm to determine, on each scan line, the boundaries of colour segments associated with a polygon.

3.6 Edge Data

The scene quantities required by the scan line algorithm for image synthesis are the polygon edges. These are formed from the perspective polygon descriptions, on the basis that the raster display comprises 575 active scan lines and each scan line is spanned by 512 pixels. Then an edge with vertices $v_1(i_1, j_1)$, $v_2(i_2, j_2)$ is characterised for the purposes of the scan line algorithm by the following quantities:

j_{min} — min (j_1, j_2): first scan line to encounter edge

j_{max} — max (j_1, j_2): last scan line crossed by edge.

i_{min} — pixel at which edge intersects j_{min}

Δi — change in edge intercept per scan line.

$EN\ EX$ — a one bit specification for edge orientation: namely if scan line enters the polygon at the edge then $EN\ EX$ is set to 1, else it is zero.

PRIORITY — an integer that specifies the polygon priority associated with the edge.

PRIORITY points to the entry in the proximity ordered list of polygon indices (PRIORITY LIST of Section 3.3) and hence links an edge to the relevant polygon and its associated features such as colour. The six lists will be collectively referred to as the EDGE LIST. It should, however, be noted that in addition to the above quantities in the EDGE LIST, the aforementioned PRIORITY LIST together with an indexed list of polygon features has to be passed to the scan line algorithm as each new image is synthesized.

A hardware realization (excluding edge smoothing and shading) of the scan line algorithm utilizing fixed point arithmetic will require the following minimum data lengths for the above edge descriptors:

j_{min}, j_{max} — 10 bits (integer)

i_{min} — 18 bits (9 bit integer + 9 bit fraction)

Δi — 18 bits (9 bit integer + 9 bit fraction)

$EN\ EX$ — 1 bit

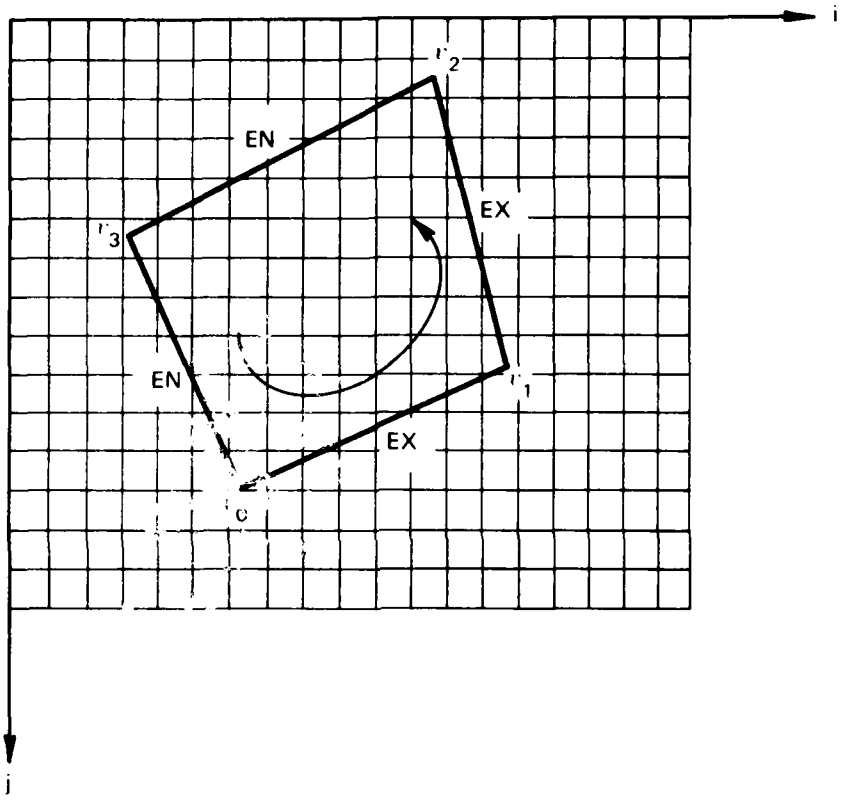


FIG. 3.6 ILLUSTRATION OF "ENTER" (EN) AND "EXIT" (EX) ORIENTATION OF POLYGON EDGE.

PRIORITY int ($\log_2(m-1)$ + 1) bits; where m is the maximum number of visible object faces and int ($\cdot$) is the truncation operator, i.e. int (l) is the greatest integer less than l.

Denoting by i_k the intercept of the edge with scan line $j = k$, the recursion

$$i_{k+1} = i_k + \Delta i; \quad k = j_{min}, \ldots, j_{max}; \quad i_{k=j_{min}} = i_{min}$$

gives the edge intercept with subsequent scan lines, with the 9 bit integer part of result being utilized by the display hardware.

3.7 Horizon Synthesis

For a scene observed near the surface of the earth the horizon can be represented, in the environment coordinate frame, by a circle in the earth plane, with centre along the line perpendicular to the earth plane, passing through O and with radius

$$r = \sqrt{2hr_e} : h/r_e \text{ small}$$

where r_e is the radius of the earth and h is the altitude of the observer.

Recall (Appendix 1) that the sequence of rotations that transforms the scene from environment to observer coordinate frame is

(i) θ_1 about $O\hat{Z}$ axis

(ii) θ_2 about rotated OY_2 axis

(iii) θ_3 about rotated OX axis

where $O\hat{X}\hat{Y}\hat{Z}$ is the coordinate frame with origin coincident with O, and parallel to the environment coordinate frame. The first rotation has no effect on the horizon image, whilst θ_2 determines the horizon line image as shown in Figure 3.7(b), where δ is the horizon depression (Fig. 3.7(a)), d is the perpendicular distance of the screen from the observer, and O_{im} is the image of O. Then the third rotation, θ_3, defines the slope of the horizon line (Fig. 3.7(c)) such that in the $O_s X_s Y_s$ coordinate frame its equation is

$$y_s = (\tan \theta_3)(x_s - \bar{x}) + \bar{y} = p/\cos \theta_3; \; p = d \tan (\delta + \theta_2)$$

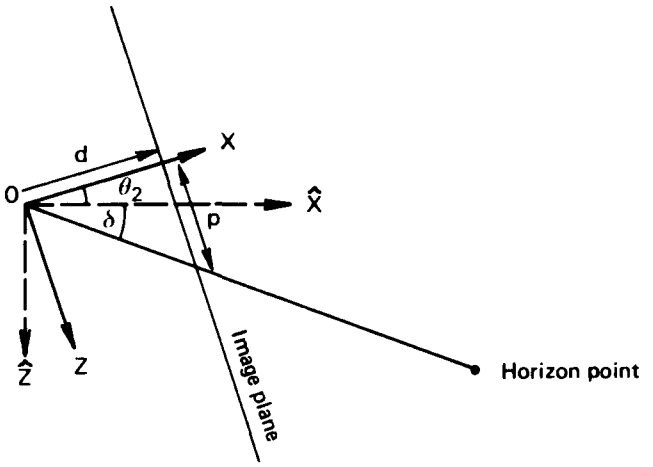
with p illustrated in Figure 3.7.

Clipping this line against the display screen boundaries, and with due consideration given to the magnitude of θ_3, the image of the earth plane is defined by a polygon with one edge the horizon and the remaining edges coincident with the screen boundaries. For convenience it is assumed that the polygon defined by the display screen represents the sky with the earth and other features superimposed upon it. To accommodate this within the proximity ordered PRIORITY LIST, the sky polygon is assigned the highest face priority index with the earth polygon being assigned penultimate priority index. The edges derived from these two polygons are treated in exactly the same way by the scan line algorithm as edges from other environment features.

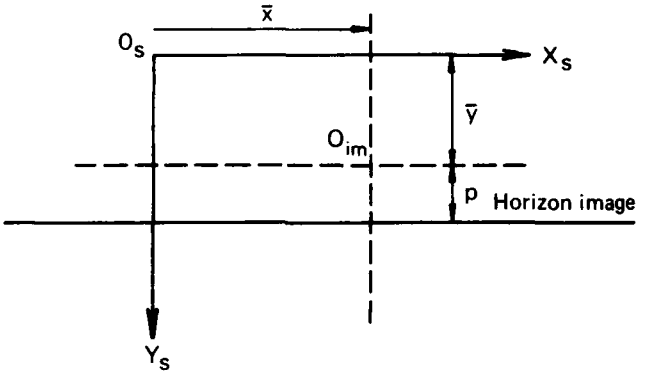
4. SCAN LINE ALGORITHM

To complete the image synthesis process, the edge data in the EDGE LIST has to be transformed to provide the colour information at each pixel as required by the display hardware. In the present case this is accomplished by a scan line algorithm which sorts the edges top-to-bottom with respect to the screen, left-to-right on each scan line and determines the visible surface at each pixel by depth comparison. The two features of the algorithm that facilitate this are (i) a one-bit indexed memory list, updated along the scan, that stores the surfaces crossed by the scan at every pixel on a scan line, and (ii) a "look ahead" register to provide the mechanism to decide whether or not an edge presents a new visible surface on the scan.

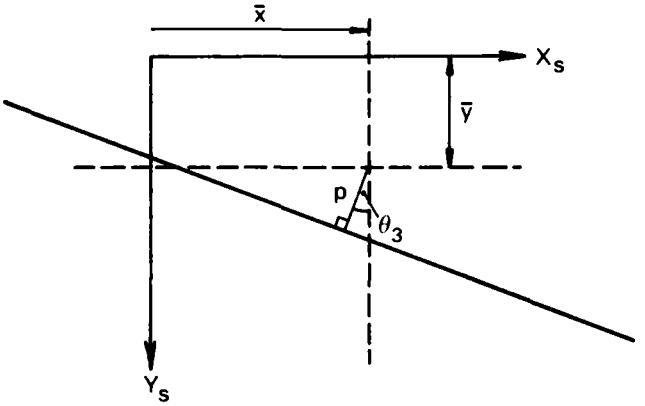
The algorithm utilizes the coherence properties of the image [1], [6], to reduce computation by presenting the sorting routines with partially ordered lists as inputs. That is, the sorted lists



(a) Horizon geometry in the plane of the rotated OY, OZ axes



(b) Image of horizon after rotation θ_2.



(c) Find image of horizon after notation θ_3.

FIG. 3.7 HORIZON GENERATION.

pertaining to scan line n form part of the data for scan line $n + 1$, and since the image is relatively invariant from one scan line to the next, the computational effort in sorting is minimised. This implies that since there is little difference in efficiency of sorting methods for lists nearly in-sort [6], the choice of a sorting method is to be dictated by ease of implementation. In the present software approach a bubble sort and a tree sort were tested, with no discernible difference, and as a result the shorter code length of the bubble sort algorithm resulted in its adoption.

Functionally the scan line algorithm consists of three distinct phases: Frame Phase, Line Phase and Element Phase, each of which are discussed in detail below.

4.1 The Frame Phase

The primary functions of the Frame Phase are the initialization of certain lists and the creation of the INCIDENCE LIST.

At the start of each image update (or frame), the scan line algorithm sorts the list of EDGE LIST indices in ascending order of j_{min}, such that the edges in the image appear in the list in the order in which they are encountered by the scan. This sorted list is termed the INCIDENCE LIST.

In addition to the INCIDENCE LIST, several "pointers" have to be initialised in this phase. Firstly, prior to the initial invocation of the Line Phase, it is necessary to reset the size of the ACTIVE LIST of edges crossed by a scan line to zero, secondly a pointer to the INCIDENCE LIST must be set to correspond to the first element of the list, and thirdly, since the output of the scan line algorithm is two lists, or fields, the size of these fields must be set to zero at the beginning of each frame. Once these functions have been completed, image synthesis is continued by the Line Phase of the scan line algorithm.

4.2 Line Phase

For each scan line in the image, the line phase performs three functions. First, an ACTIVE LIST of edges intersected by the current scan line is computed, and secondly, for each intersected edge the intersect with the scan line is compiled. Finally the ACTIVE LIST is sorted in increasing order of edge/scan line intersect. The ACTIVE LIST is set up as a list of indices from the EDGE LIST which correspond to the intersected edges. Also, by virtue of line-to-line coherence, it is not necessary to form a new ACTIVE LIST *ab initio* for each scan line. Instead, the ACTIVE LIST for the current scan line is formed from the ACTIVE LIST of the preceding scan line. This makes ACTIVE LIST generation economical in terms of computation time.

Specifically, the ACTIVE LIST for the n-th scan line is formed from the ACTIVE LIST of the $(n - 1)$-th scan line by

(i) deleting any edge for which

$$n > j_{max}$$

(ii) appending any edge from the INCIDENCE LIST for which

$$n = j_{min} .$$

Two special cases are recognized:

(i) At the first scan line of a frame the ACTIVE LIST is initially empty. Therefore, step (ii) of the above procedure suffices for the generation of the ACTIVE LIST.

(ii) If the ACTIVE LIST from the previous line only contains one entry, the deletion routine is again skipped. This is because the ACTIVE LIST for every scan line necessarily contains one entry for the sky as discussed in Section 3.7.

Deletion of entries from the ACTIVE LIST

The ACTIVE LIST is scanned until either (i) the end is reached, or (ii) an entry that should be deleted, is detected. In case (i), there is nothing to be done, so control passes on to the append

routine. Alternatively, in case (ii) the entry is deleted and the next valid entry is transferred to the vacated location in the ACTIVE LIST. The flow chart for the deletion routine is illustrated in Figure 4.1.

For the edges that remain in the ACTIVE LIST, the intersections with the scan line are computed by incrementing the intersects of the preceding line with the respective edge gradients, viz.

$$i_{min} = i_{min} + \Delta i$$

as referred to in Section 3.6. Note that since the active list only points to edge entries in the EDGE LIST, the recurrence relation refers to the i_{min} entry of the relevant edges.

Appending entries to the ACTIVE LIST

All that remains to be done to complete the ACTIVE LIST update is to append any new edges encountered by the scan. The mechanics of this is as follows: during each frame a pointer is maintained indicating the entry of the INCIDENCE LIST to be processed next. If j_{min} corresponding to this entry is equal to the scan line number, then this edge is added to the ACTIVE LIST and the INCIDENCE LIST pointer is incremented. The procedure is repeated until the equality is false, at which time the ACTIVE LIST for the scan line is complete.

Thus, by virtue of the INCIDENCE LIST being ordered with respect to j_{min}, the pointer steps through the list appending entries to the ACTIVE LIST at the appropriate scan lines. The flow chart for the routine is shown in Figure 4.2. Note that for the appended edges, the intersections with the scan line are just the values of i_{min}, computed during edge generation, and stored in the EDGE LIST.

The final function of the Line Phase is to sort the ACTIVE LIST in increasing order of edge/scan line intersections. Since the image possesses line-to-line coherence, the ACTIVE LIST will be nearly in-sort, hence minimizing the computational effort for sorting. The linked structural relationship between the EDGE LIST, INCIDENCE LIST and ACTIVE LIST is illustrated in Figure 4.3.

4.3 The Element Phase

The purpose of this phase is to solve the hidden surface problem by determining the visible surface at each pixel on the current scan line. The essential feature to be exploited in achieving this is pixel-to-pixel [7] coherence. That is along a segment of scan line bounded by two edges only one surface is visible, and hence it suffices to establish the visible edges along each scan line. For this purpose the following additional data structures are required:

(i) a list parallel to the PRIORITY LIST discussed in Section 3.3, termed the ON/OFF LIST. Generated at each scan line, the ON/OFF LIST contains one entry for each surface in the image (as does the PRIORITY LIST), and like the PRIORITY LIST, the ON/OFF list may be referenced via the priority entry in the EDGE LIST. The function of the list is to indicate the current position of the scan relative to each surface crossed by the scan line. That is, if the scan crosses an ENTER edge of a face, the ON/OFF entry corresponding to the face is "turned on", and conversely if the edge is an EXIT edge, it is "turned off". At the beginning of each scan line the list is initialized to "off".

(ii) Two lists referred to as FIELD 0 and FIELD 1 (with associated pointers) that store output of the scan conversion process. The two lists correspond to the two fields of the interlaced display hardware, and are more fully discussed in Section 4.4.

(iii) Three registers:

(a) A current face register, CR, loaded from the ACTIVE LIST, and which points to the EDGE LIST entry corresponding to the edge being processed by the element phase.

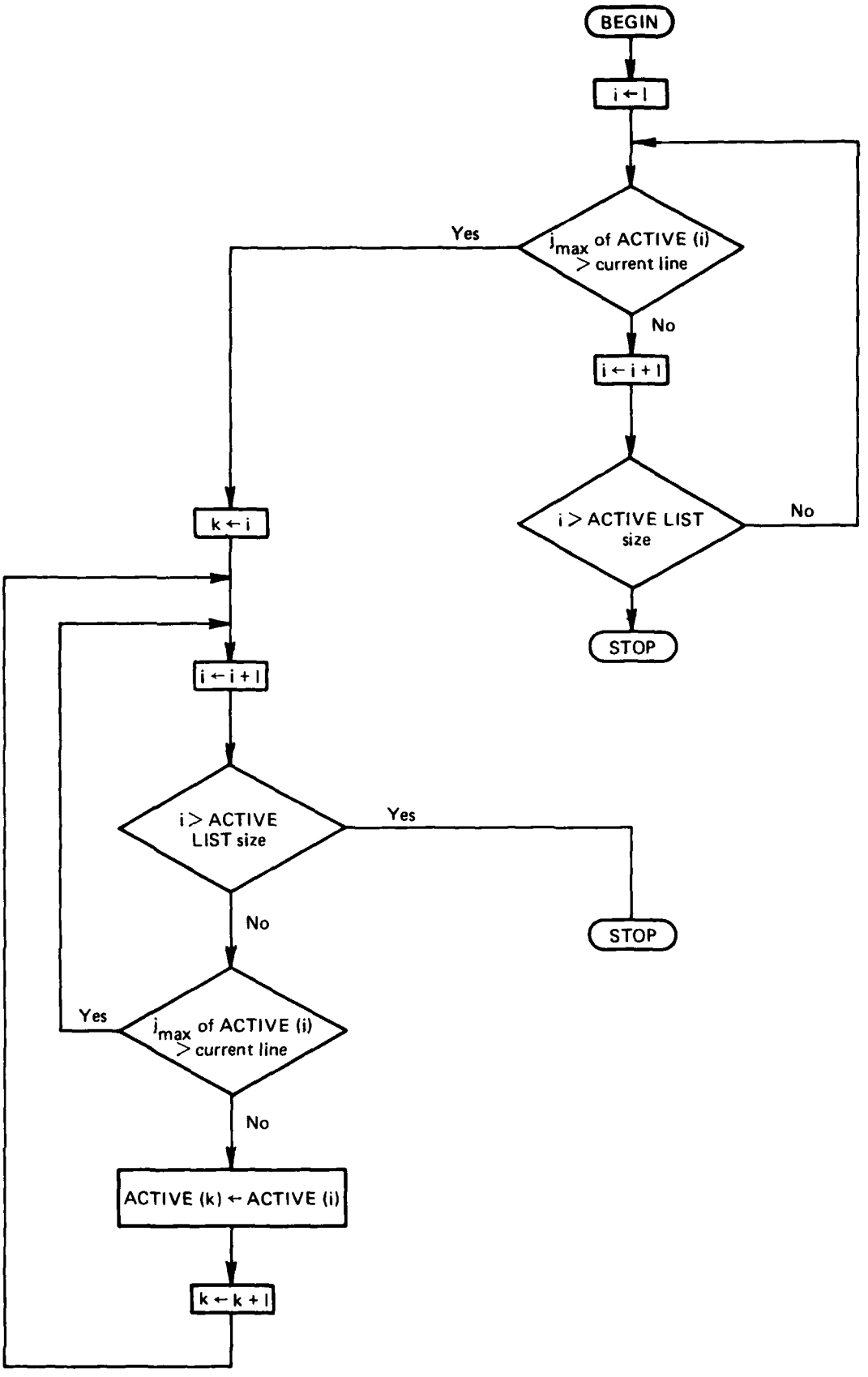


FIG. 4.1 DELETION ROUTINE

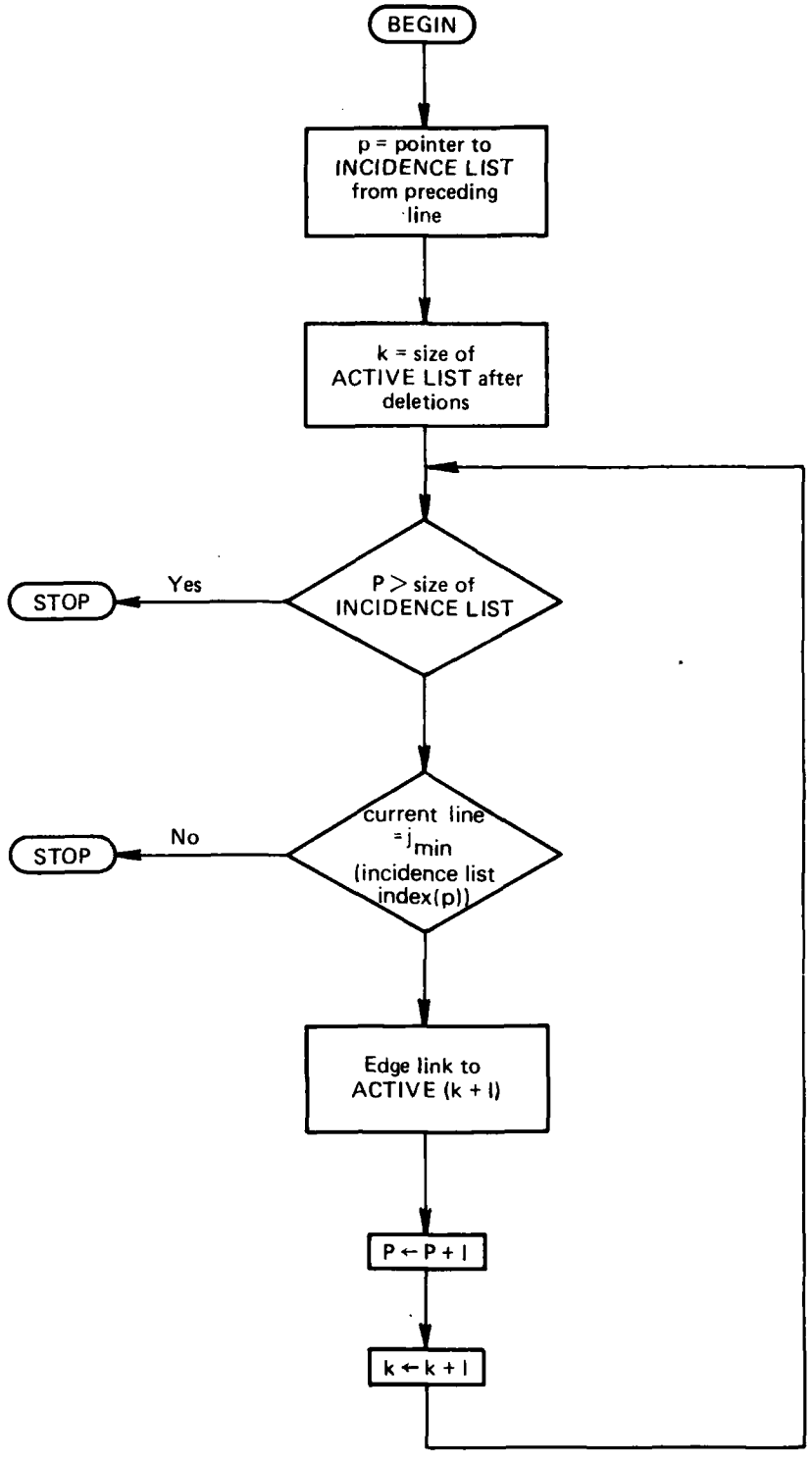


FIG. 4.2 APPEND ROUTINE

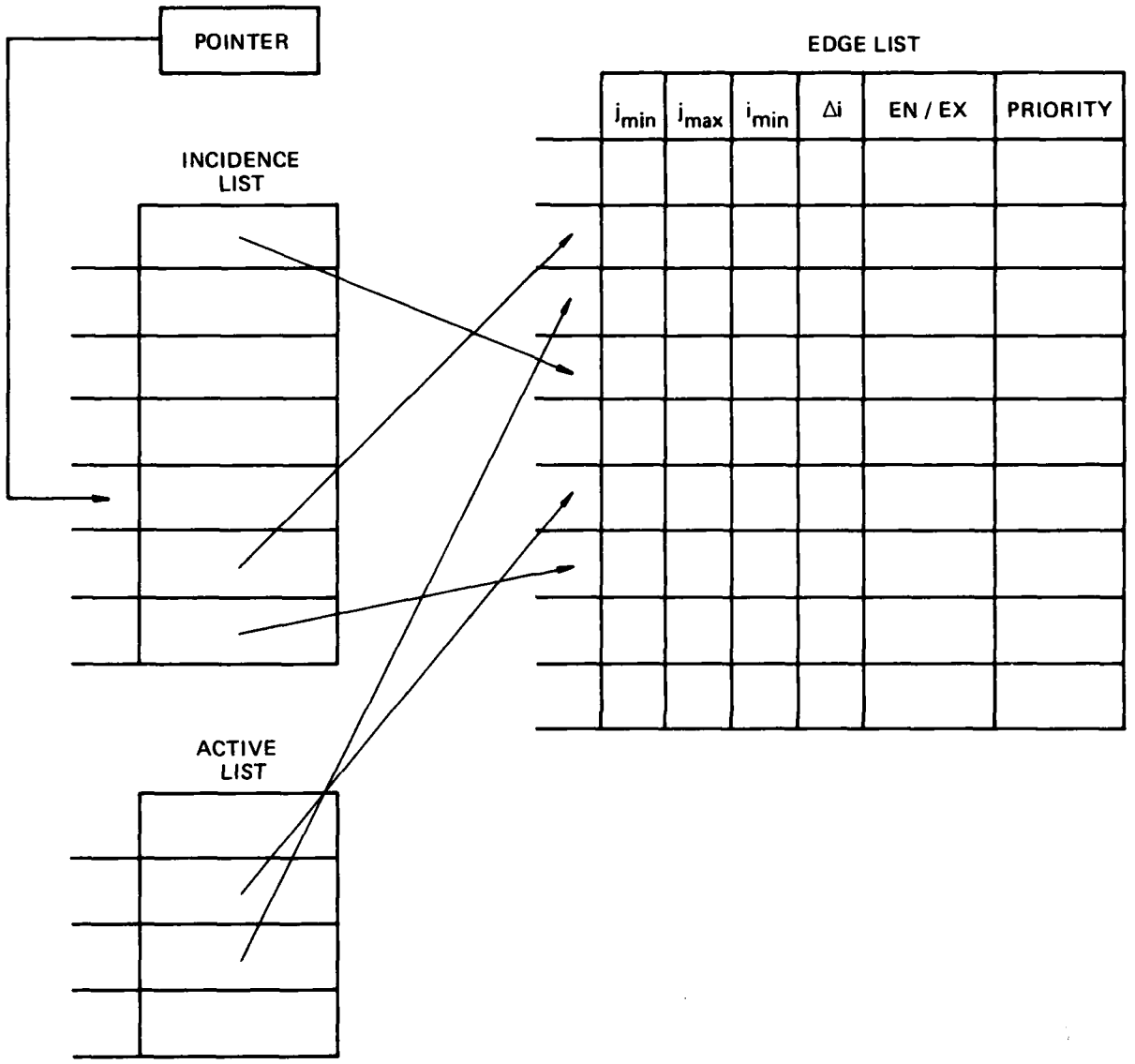


FIG. 4.3 STRUCTURAL RELATIONSHIP BETWEEN EDGE INCIDENCE AND ACTIVE LISTS.

(b) A marker register, MR, loaded from the ACTIVE LIST, and which points to the EDGE LIST entry corresponding to the next edge to be processed.

(c) A backup register, BR, loaded from the PRIORITY entry of the EDGE LIST, and which stores the PRIORITY of the face visible at the pixel immediately preceding the intersection of the scan line and the edge referred to by CR.

The algorithm employs the MR register to perform one edge "look ahead" whilst processing the edge indicated by CR. It compares the i_{min} values of the edges referred to by CR and MR and branches to either the MULTIPLE EDGE routine if the values are equal or the ENTRY/EXIT routine if they are unequal. The former condition, termed an "image singularity" arises when multiple edges cross a scan line at the same pixel, and hence requires additional effort to establish priority.

The data structure for the Element Phase is illustrated in Figure 4.4.

The ENTRY EXIT routine

This is the routine invoked at object silhouette edges. First the CR register is examined to determine via the EDGE LIST whether the edge referred to is an ENTER or an EXIT edge.

(i) ENTER edge.

First, the flag, in the ON/OFF LIST, corresponding to the edge indicated by CR is turned 'on'. Next the visibility of this edge relative to the face referred to by BR has to be established by comparison of the face priorities. If the face referred to by CR has lower PRIORITY value (i.e. closer to the observer) than the face referred to by BR, then BR is reset with the new face PRIORITY, and the values of i_{min} and face COLOUR corresponding to CR are appended to either FIELD 0 or FIELD 1 depending on the scan line number.

(ii) EXIT edge.

The flag referred to by CR in the ON/OFF list is turned OFF and the PRIORITY of the faces corresponding to CR ad BR are compared. If they are not equal, then the face referred to by BR remains visible and the edge of CR is occulted by it. On the other hand, if they are equal, the ON/OFF LIST is scanned to determine the face of the lowest PRIORITY value with ON/OFF entry set to 'on'. The PRIORITY of this face is registered in BR, and its colour together with i_{min} of the edge in CR is appended to one of the output lists. Note that by virtue of the fact that the "background" has highest PRIORITY value and its flag is always set 'on', no degenerate situations can arise.

At the conclusion of the ENTRY/EXIT routine, MR is copied into CR, the EDGE LIST index of the next edge intercept is copied from the ACTIVE LIST into MR and the cycle starts again. The process continues until the ACTIVE LIST is exhausted.

As an example consider the situation depicted in Figure 4.5, where it is assumed that face #2 is closer to the observer than face #1. Further assume that the routine has progressed to the edge that intersects the scan line at i_3, where it is desired to determine which surface is visible. The content of the registers, ACTIVE LIST and the ON/OFF LIST are as depicted in Figure 4.6.

At i_3, the CR holds the link to the first edge in the EDGE LIST. From the EN/EX it can be seen that the scan enters face #2 and hence turns on the first entry of the ON/OFF since face #2 has PRIORITY equal to zero. Moreover since the PRIORITY of this edge is less than the PRIORITY of the face corresponding to BR, the registers are modified as in Figure 4.7.

At pixel i_4 (correponding to CR) the scan crosses the EXIT edge with PRIORITY equal to zero, and hence the first entry of ON/OFF is set to 'off'. Moreover since BR points to the same face as CR, the algorithm scans down the ON/OFF list to find the first entry set 'on', which in this case happens to be the face with PRIORITY equal to 1. The BR register is set with this value, and the registers have the contents shown in Figure 4.8.

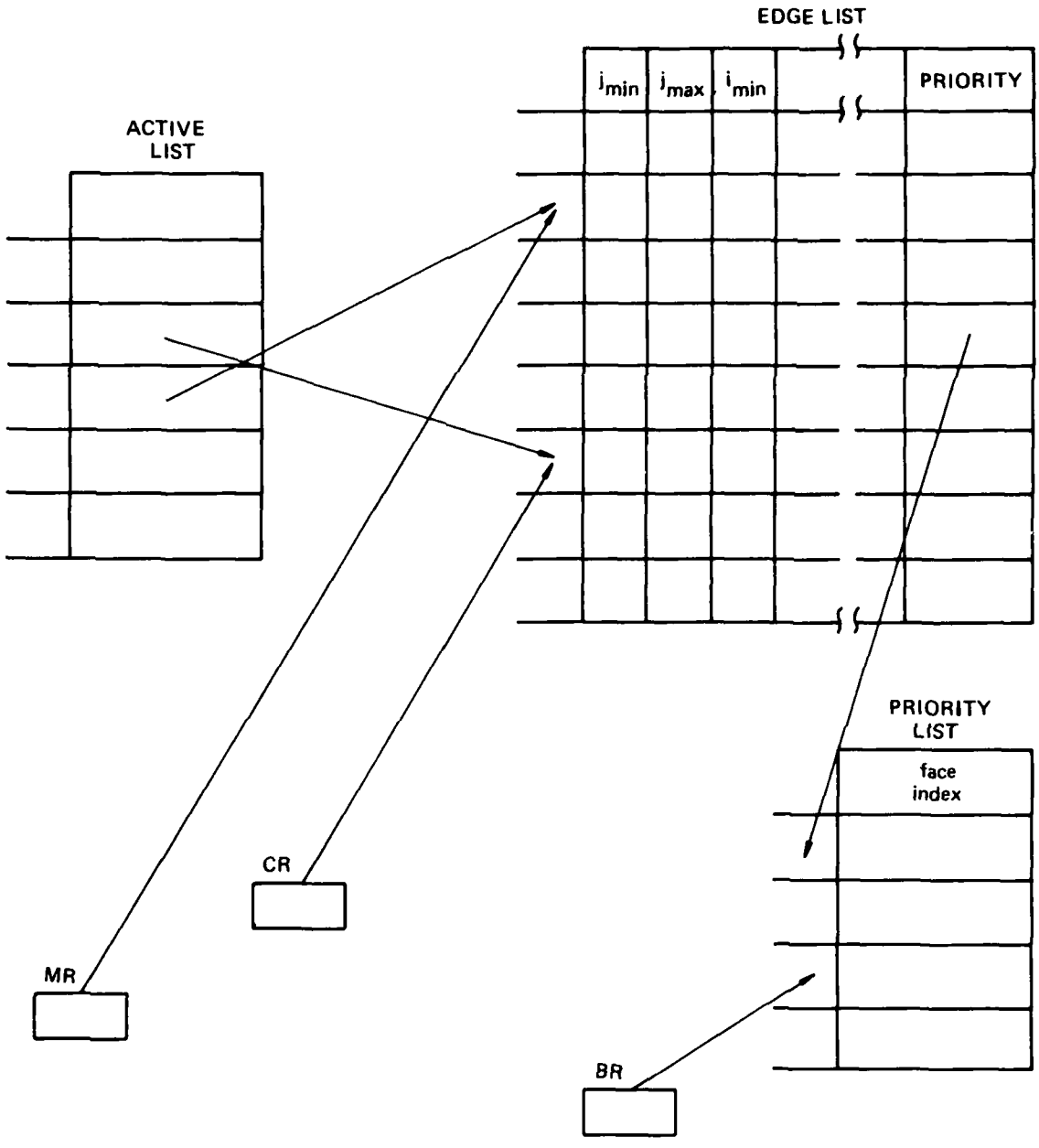


FIG. 4.4 DATA STRUCTURE FOR ELEMENT PHASE.

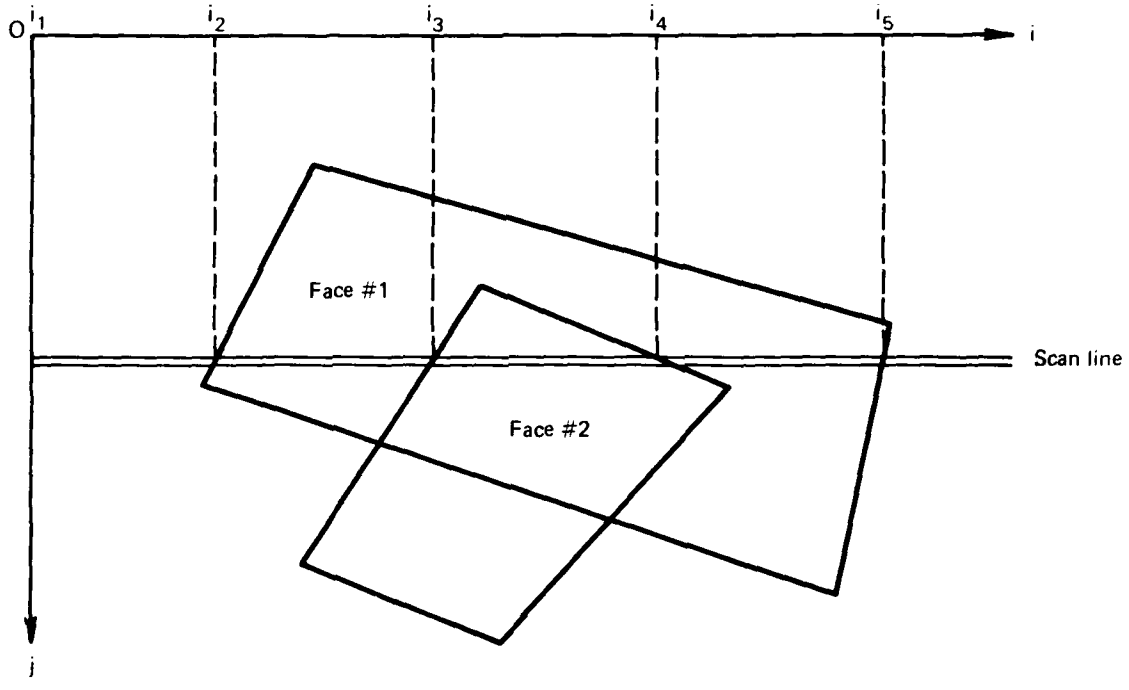


FIG. 4.5 EXAMPLE OF TWO SURFACES REQUIRING THE SOLUTION OF THE HIDDEN SURFACE PROBLEM.

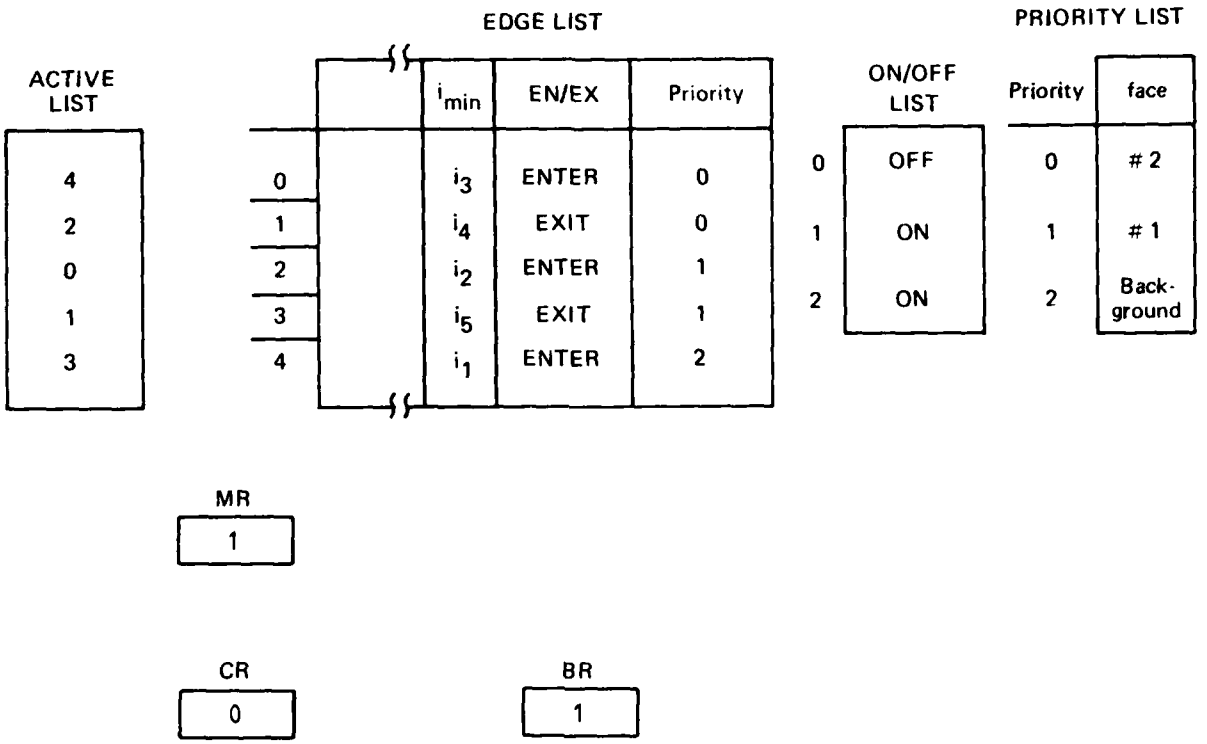

EDGE LIST
PRIORITY LIST
ACTIVE LIST
ON/OFF LIST
i_{min}
EN/EX
Priority
Priority
face
4
2
0
1
3
0
1
2
3
4
i_3 ENTER 0
i_4 EXIT 0
i_2 ENTER 1
i_5 EXIT 1
i_1 ENTER 2
0 OFF
1 ON
2 ON
0 # 2
1 # 1
2 Back-ground
MR
1
CR
0
BR
1

FIGURE 4.6

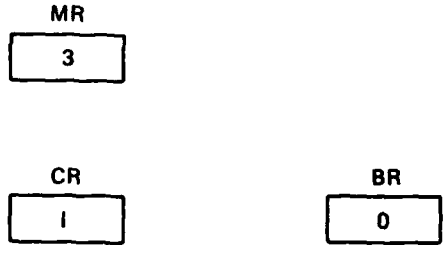


FIGURE 4.7

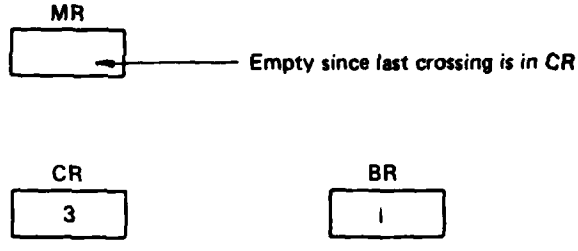


FIGURE 4.8

A functional flow chart of the ENTRY/EXIT routine is illustrated in Figure 4.9.

The Multiple Edge Routine

This routine is invoked when two or more successive entries in the ACTIVE LIST have identical entries for i_{mtn} (i.e. more than one edge intersects the scan line at the same pixel). The situation arises due to the finite resolution of the display and is typified by

(i) two abutting faces as illustrated in Figure 4.10(a)

(ii) the vertex formed by two edges as in Figure 4.10(b)

(iii) the perspective image of a vertex of a three dimensional object as in Figure 4.10(c).

The decision as to which surface is visible at the pixel is complicated by the fact that the ENTER and EXIT edges belonging to a given face are not in any particular order with respect to themselves in the ACTIVE LIST. Thus processing, say case (ii), by the ENTRY/EXIT routine could result

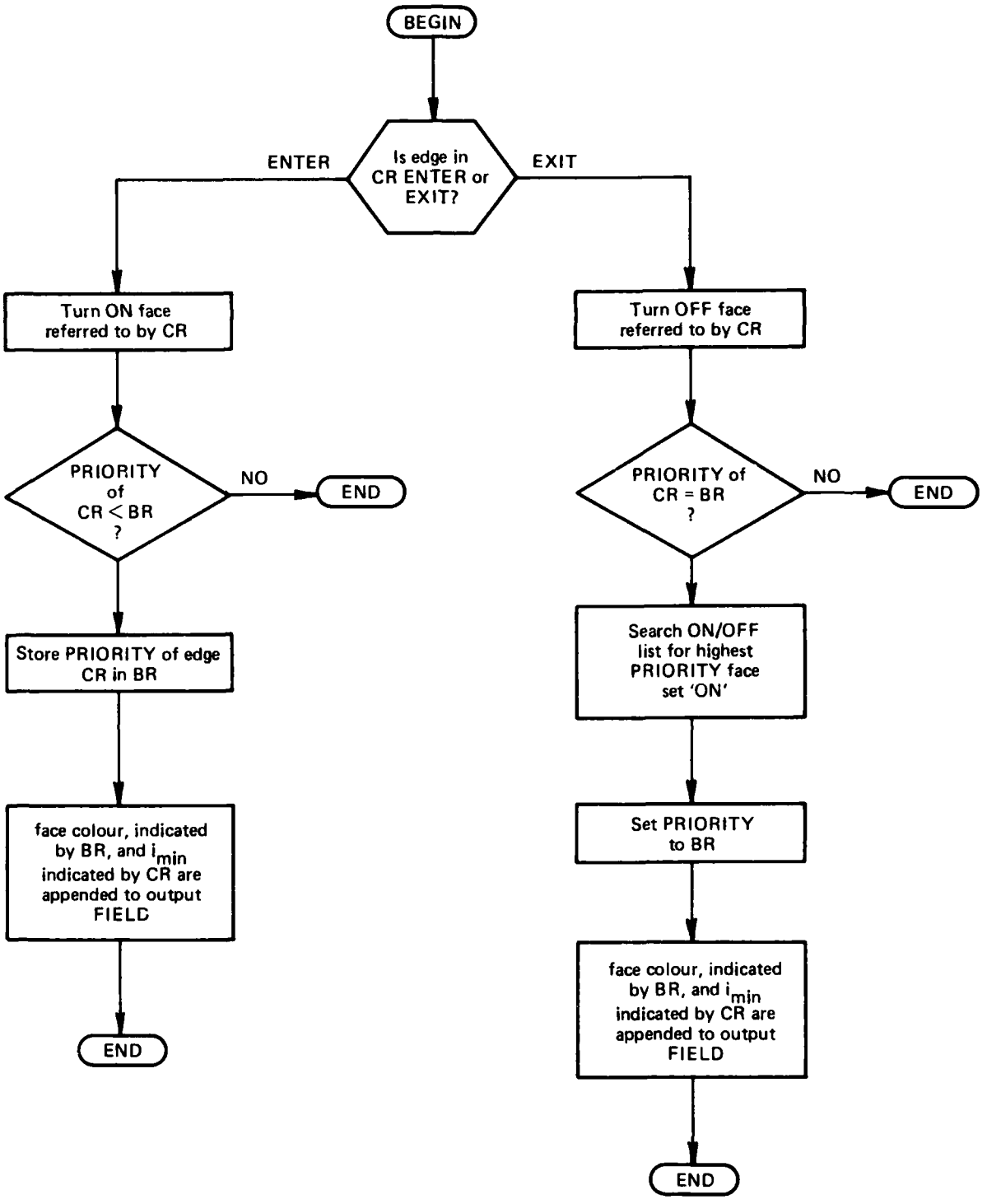


FIG. 4.9 FUNCTIONAL DIAGRAM OF ENTRY/EXIT ROUTINE.

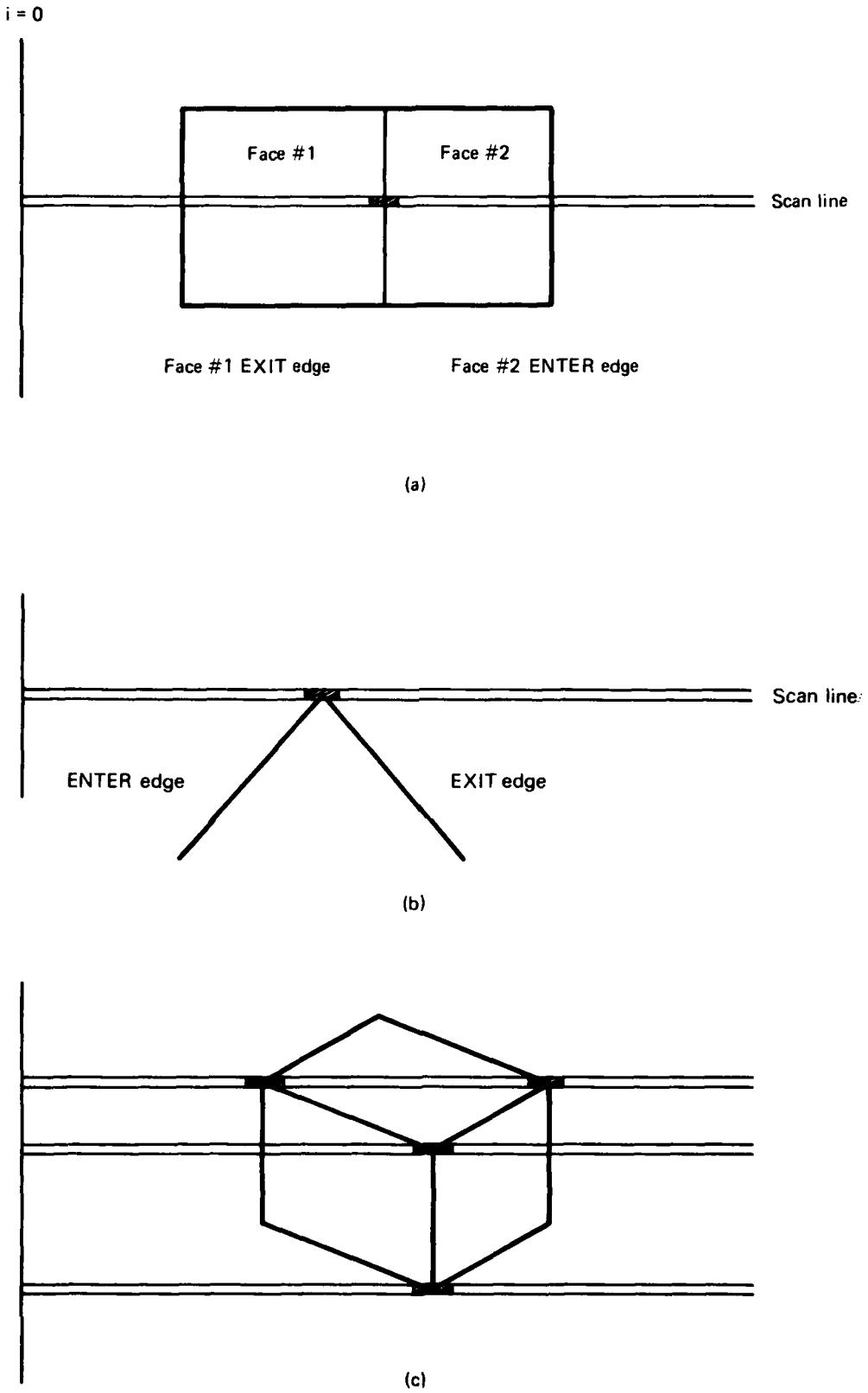


FIG. 4.10 MULTIPLE EDGE CROSSINGS

in the surface being first turned OFF and then turned ON, if the OFF edge precedes the ON edge in the ACTIVE list.

The above problem is solved by the MULTIPLE EDGE routine. It scans twice, those entries of the ACTIVE LIST which refer to the same pixel. On the first pass those entries which are ENTER edges (as indicated by EN/EX entry in EDGE LIST) have their corresponding surfaces turned 'on' in the ON/OFF list. On the second pass those entries which refer to EXIT edges, have the corresponding surfaces turned 'off'. The ON/OFF list is then searched to ascertain which of the 'on' surfaces has the lowest PRIORITY. If this is different from the contents of BR, then BR is set to the new value, and the pixel number referred to by CR together with the surface colour of BR are appended to the relevant output FIELD. If the PRIORITY is equal to BR then the preceding steps are bypassed, and MR and CR are updated from the ACTIVE LIST and execution returns to the beginning of the Element Phase.

4.4 Experimental Interface

Although the principal topic of interest here has been the CGI process from the point of view of image data and scan conversion algorithms, it is appropriate at this point to mention the salient features of the experimental CGI display interface.

The output of scan conversion are two lists or fields, corresponding to an interlaced raster scan. Each field (FIELD O, FIELD I) comprises a sequential list (ordered for left-to-right, top-to-bottom scanning) of edge/scan line intersection and surface colour to the right of each intersection. It is worthwhile noting that since the display screen is treated as the background surface by the CGI algorithm, there is for every line an entry, in the appropriate field, corresponding to $i_{min} = 0$. This entry then constitutes a delimiter between succeeding scan lines precluding the necessity to tag any of the data with scan line identifiers.

Functionally, the interface hardware comprises a 16-word deep first-in, first-out (FIFO) memory that buffers the field data, a nine bit (equivalent to 512 pixels per scan line) pixel counter driven from a 10 MHz clock synchronised to the composite video signal driving the raster display, and a 9 bit comparator.

In operation, the 9 bit intersection data in the bottom word of the FIFO is compared with output of the pixel counter, and on equality the colour information at the bottom of the FIFO selects the colour to be displayed along the scan onwards from the pixel indicated by the counter. The contents of the FIFO are pushed down and a new data entry from the output FIELD is placed at the top of the FIFO. The comparison and, as appropriate, the shift down process continues until the end of the scan line. The counter is reset to zero at the end of each line, the comparator is enabled at the deginning of the next line and a faithful synthesis of the image results by virtue of the implicit line delimiting function of the left edge of the screen as described above. A functional diagram appears in Figure 4.11.

Note, in the above context, the importance of clipping and conversion of data to pixel/scan line units. If an intercept, i_{min}, along a scan line is greater than 511, then the comparator will not be in coincidence with the counter and the image may have gross aberrations. Similarly, assuming 575 active scan lines (287 in one field and 288 in the other), an incorrect number of entries in either output FIELD will cause interlace problems manifested as a disjoint image.

4.5 Algorithm Evaluation

There are two aspects to the assessment of the scan line algorithm. First, the capability of the algorithm to determine the visible surface at each pixel, and hence produce a valid image, has to be verified. Secondly, in light of the real time performance requirement for flight simulation, it is of interest to determine indicative estimates of the relative computational effort required by the algorithm in the context of the CGI process.

For these purposes the algorithms comprising the image generation process were implemented in software on the ARL DECsystem-10 computer. Frame and Line Phase routines of the scan conversion algorithm were programmed in assembly language, whilst the Element Phase together with the algorithms related to edge generation were programmed in Fortran. The test environment comprised a rectangular prism and a tetrahedron positioned close to

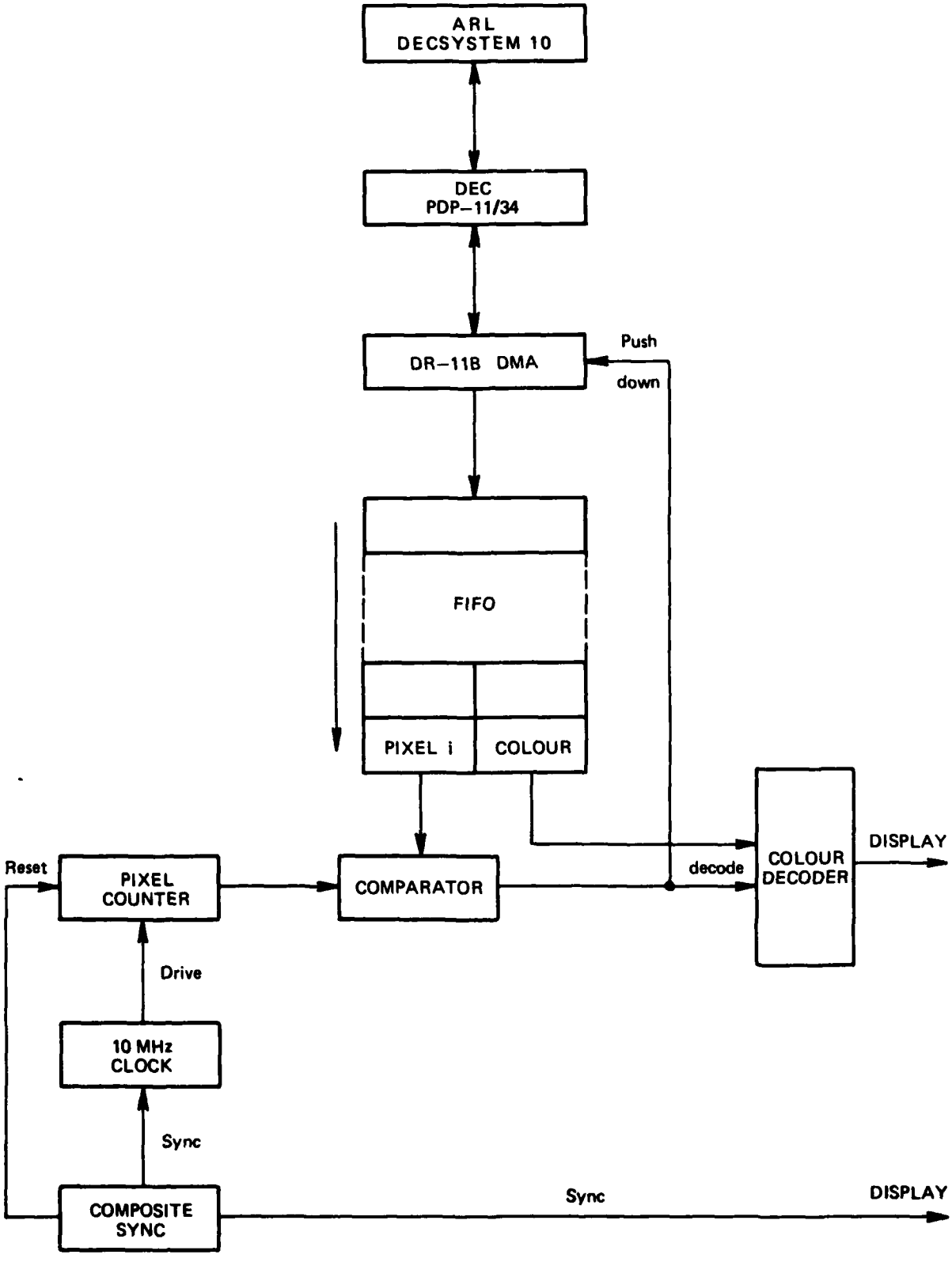


FIG. 4.11 FUNCTIONAL DESCRIPTION OF EXPERIMENTAL CGI–DISPLAY INTERFACE.

the origin of the environment coordinate frame. For the purpose of producing different occultations of the scene objects, the observer was assumed to move along a path encircling the objects, and maintaining a viewing direction aimed at the environment origin as illustrated in Figure 4.12.

Photographs of images computed by the DECsystem-10 and displayed on a raster display are shown in Figure 4.13 and provide visual validation of the scan line algorithm in operation. In particular it can be seen that the MULTIPLE EDGE routine resolves the visibility problems at the tetrahedron apex and at the vertices of the rectangular prism.

To provide average values of the computation load, images of the environment objects were computed for 50 equidistant viewpoints along the observer's path. With the aid of a DECsystem-10 program which samples the program counter at regular time intervals, a histogram (Fig. 4.14) was constructed which describes the areas of the CGI program in which CPU time was expended. It can be readily seen from Figure 4.14 that the scan line algorithm accounts for 90% of the computation effort to produce an image, with polygon clipping accounting for 35% of the remaining computation time. The dominance of the scan line algorithm in this respect is not surprising if one considers that Line Phase functions are executed 575 times, whilst depending on the average number of edge/scan line intersects some Element Phase functions are executed well in excess of the latter value.

Restricting attention to the scan line algorithm the relative computer utilization of different phases and functions is presented in histogram form in Figure 4.15. It is evident that in the Line Phase significant computational effort is expended in computing the edge/scan line intercepts for the ACTIVE LIST and the sorting of the ACTIVE LIST. On the other hand the computation time for the Element Phase is dominated by the ENTER/EXIT routine although it is interesting to note that the MULTIPLE EDGE routine accounts for 25% of this phase which is indicative of the extra effort required to solve the visibility problem at image singularities and their not infrequent occurrence.

In addition to the comparative measurements the total computation time for the generation of an image of the complexity illustrated in Figure 4.14 was measured (via DECsystem-10 clock) to be 500 msec, whilst the polygon clipping algorithm was measured independently of the rest of the CGI process, to take 3.9 msec per quadrilateral. The latter value is as one would expect since polygon clipping accounts for 3% of the total image computation and each image of Figure 4.13 contains 4–5 polygons.

Although the preceding quantitative results pertain to a software implementation on a general purpose computer, they are indicative of the fact that from the point of view of computational load it is the scan line algorithm which is the key factor in image generation. Hence CGI image update rates, in the context of flight simulation (i.e. at least 12 per second), will require special purpose hardware implementation of the scan line algorithm, as well as fast general purpose computing elements for such functions as polygon clipping.

5. EXTENSIONS

Two qualities that enhance CGI image quality are shading and edge smoothing. The former is necessary for two reasons; firstly to restore the curved appearance of objects represented by polygons, and secondly to simulate, in the case of extended objects, such as runways, distance fading due to presence of a diffuse atmosphere. On the other hand edge smoothing eliminates step effects along edges arising from the discrete nature of the CGI process. Methods to implement shading and edge smoothing have received considerable attention in the literature and a number of solutions have been validated. It is the intent here to take two methods that have potential in the simulation area [4] and show how they can be accommodated within the CGI structure of the previous sections.

5.1 Surface Shading

The scan line algorithm of the preceding chapter is tailored for a visual environment modelled by planar faced polyhedra. However for realistic images the algorithm must be able to accommodate the synthesis of curved surface objects. In the context of the object models assumed it may be conjectured that by increasing the number of approximating polygons the

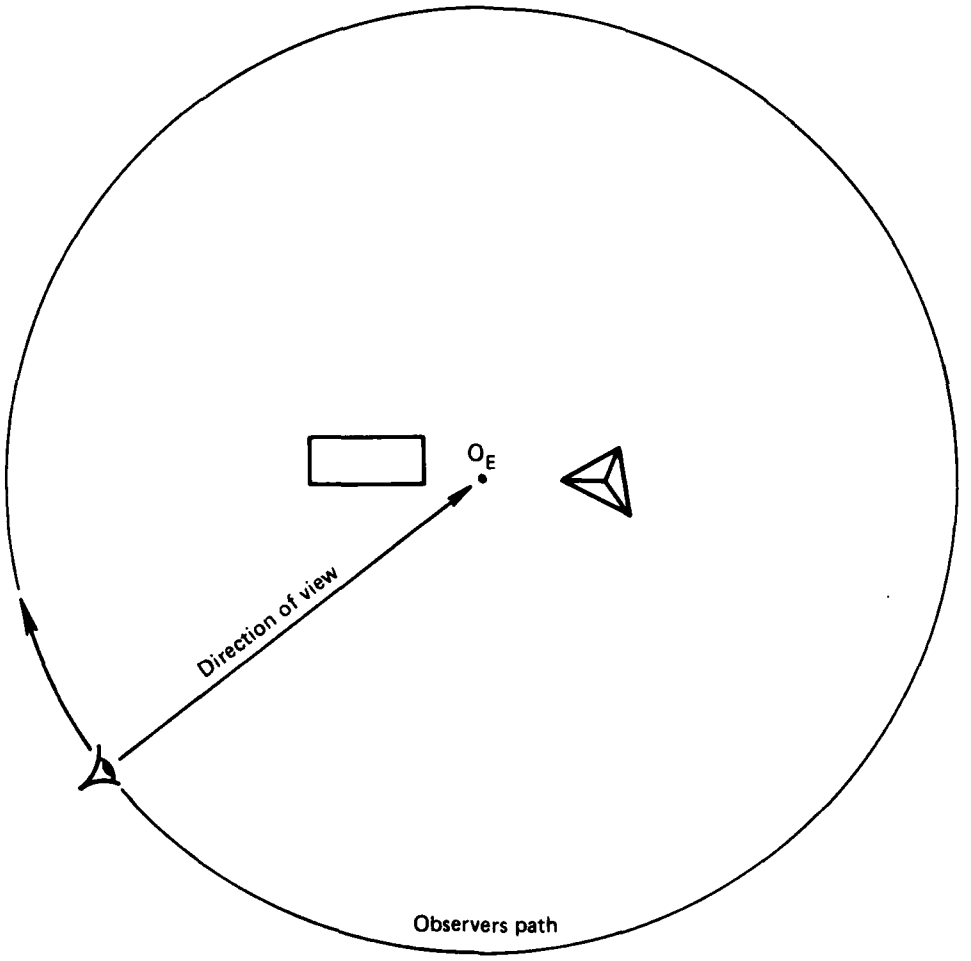
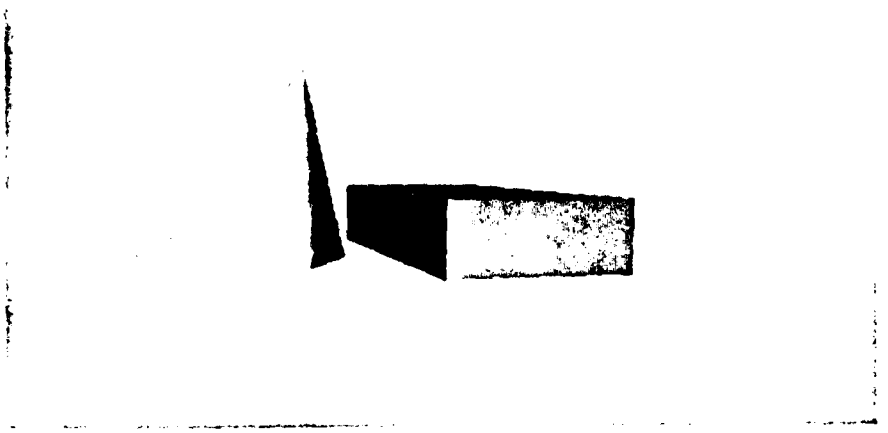
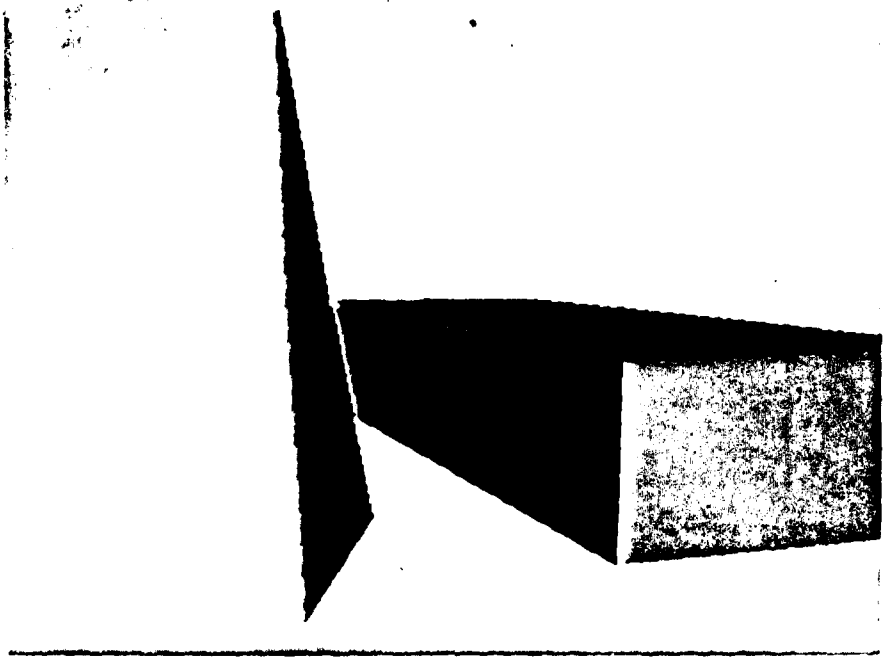


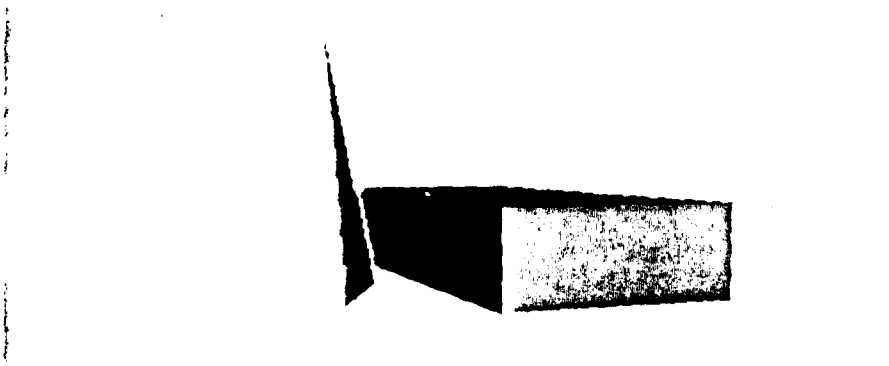
FIG. 4.12 PLAN VIEW OF TEST ENVIRONMENT AND OBSERVER'S PATH.



(a)
Test objects: *tetrahedron* and rectangular prism.



(b)
Clipping: objects clipped against top and right boundary planes.



(c)
Occultation

FIG. 4.13 TEST IMAGES

(Colour xerographic reproduction. The non-planar nature of the display screen is responsible for the apparent curvature of some of the edges.)

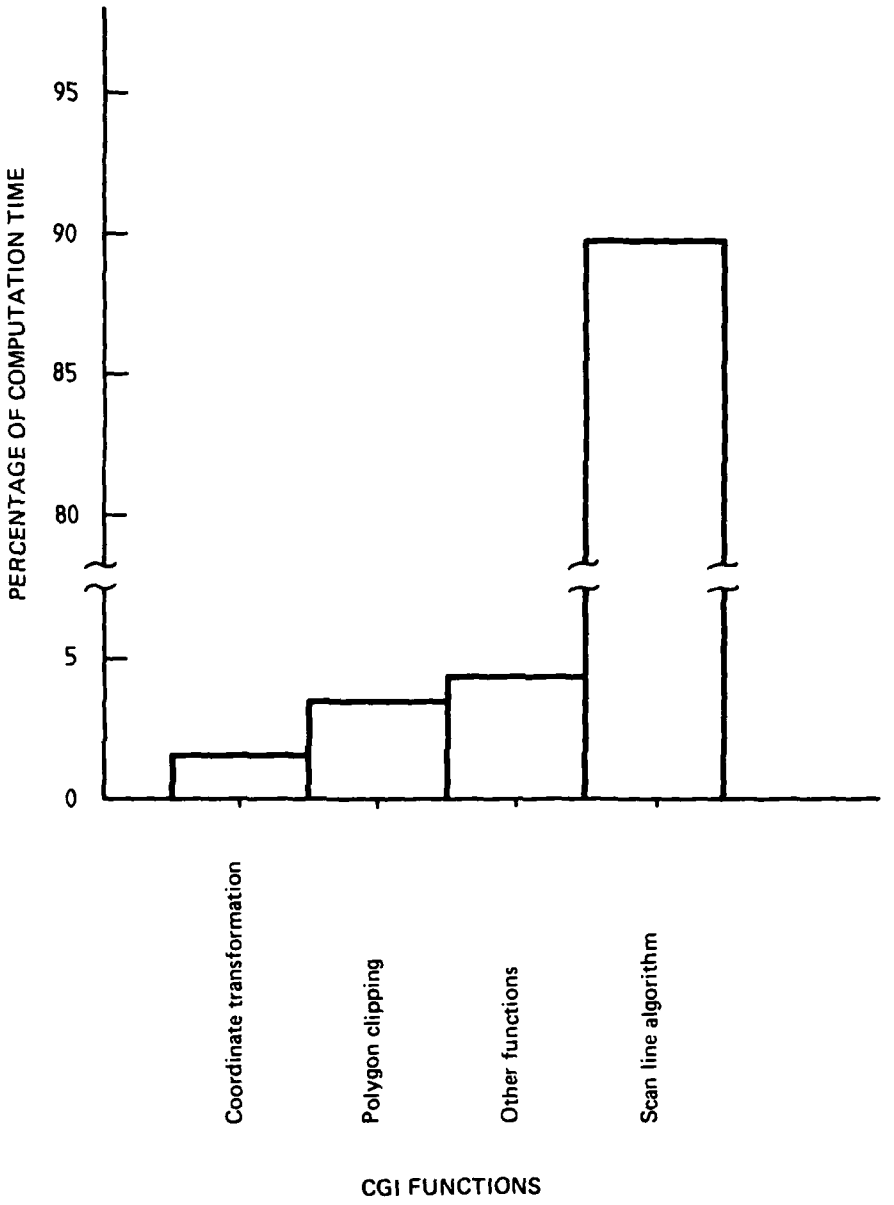


FIG. 4.14 RELATIVE COMPUTATIONAL EFFORT AS A FUNCTION OF CGI PROCESSES

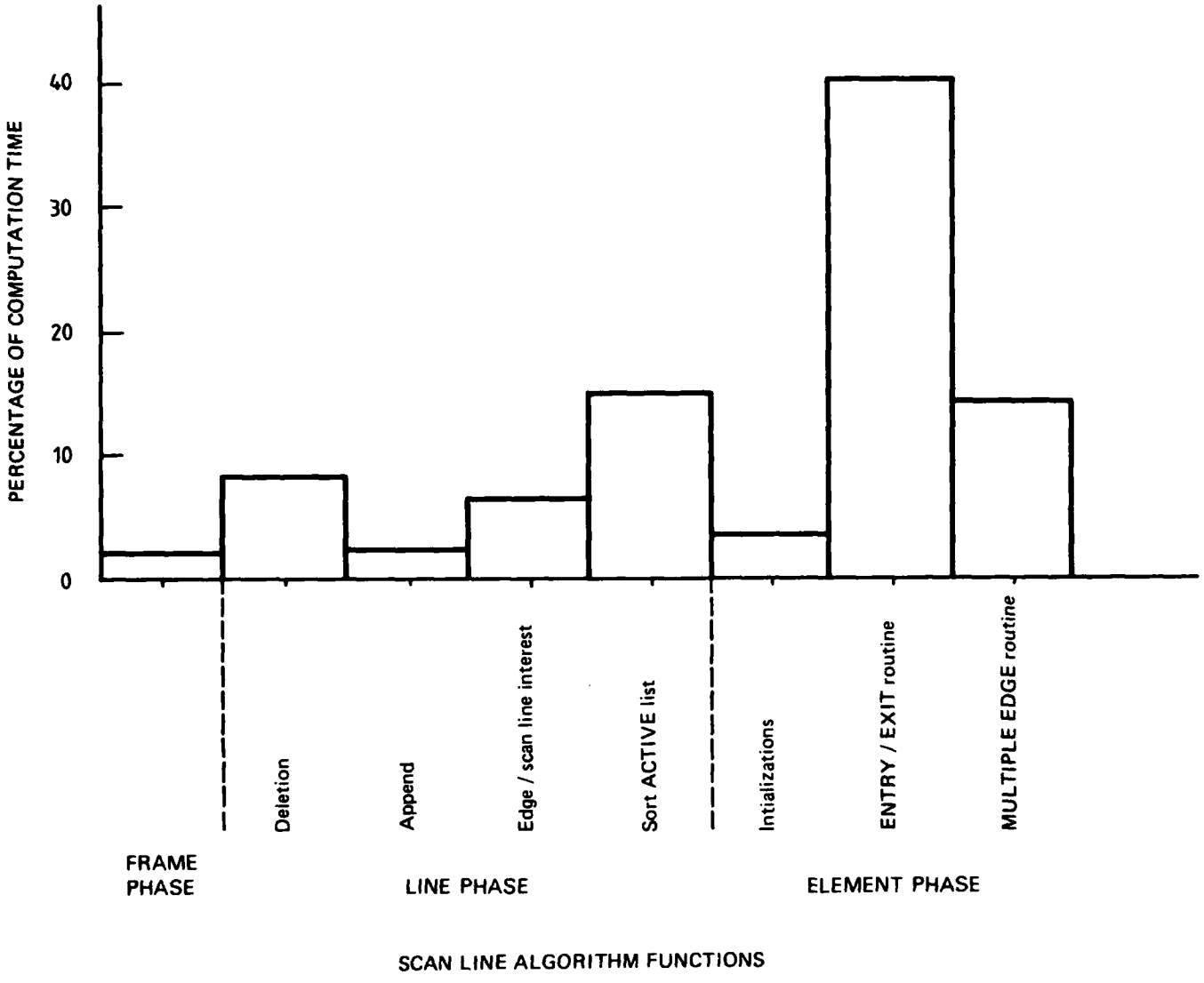


FIG. 4.15 RELATIVE COMPUTATIONAL LOAD OF THE SCAN LINE ALGORITHM.

curved appearance of an object can be restored. As outlined in [2], [4] and [5] this is not the case as psychophysical aberrations arise due to the discontinuity in colour and in the rate of change in colour across an edge.

To circumvent colour discontinuity across edges Gouraud [5] developed a method based on linear interpolation along edges and across faces to restore the smooth appearance of a curved surface modelled by polyhedra. In this approach colour is assigned to each vertex of an object face according to a shading rule, based on the distance between vertices and light source, the intrinsic colour of the surface and the relative orientation of the vertices with respect to the light source. This orientation is computed by assigning to each vertex the direction defined by averaging the normals of surfaces meeting at the vertex. The polygon is then clipped such that any new vertex created by intersection of an edge with a viewbox boundary is assigned a colour computed by interpolation using the parameter t of (3.2.3).

Although Gouraud's method results in excellent static images, aberrations may occur in dynamic pictures since the algorithm is not invariant of the relative orientation of the scan direction with respect to image polygons. To see this consider the polygon in Figure 5.1(a), where colours $c_1, \ldots, c_5$ have been assigned to vertices $v_1, \ldots, v_5$, respectively. Interpolating along edges $\overline{v_1v_2}$ and $\overline{v_3v_4}$, and then along the segment $\overline{rs}$ results in the point p being assigned colour

$$\frac{15}{49} c_1 + \frac{20}{49} c_2 + \frac{1}{14} c_3 + \frac{3}{14} c_4$$

Rotating the polygon by 90° with respect to the scan direction (Fig. 5.1(b)) the 3 step interpolation method now assigns to p colour equal to

$$\frac{9}{28} c_1 + \frac{2}{7} c_2 + \frac{2}{7} c_3 + \frac{3}{28} c_5$$

which would not yield the same result as the previous computation, except under exceptionally fortuitous circumstances.

The underlying reason for the anomaly is that Gouraud's method assumes that the coordinates and colours of image polygon vertices define a plane in the Cartesian coordinate system described by the triple (x_s, y_s, c). A sufficient condition for this to be true is that the image polygons be triangular.

In spite of the theoretical limitations of Gouraud's method it has been utilized in real time CGI [4] and hence it is worthwhile to illustrate its implementation in the framework of the present scan line algorithm. The key requirement for the method is the calculation of the following quantities for each ENTER edge

(i) $c\,(i_{min}, j_{min})$ — the colour at the vertex (i_{min}, j_{min})

(ii) Δc — the colour increment per scan line along the edge

(iii) δc — the colour increment per pixel along each scan line intersecting the face corresponding to the ENTER edge.

The term Δc is equal to $dc/dy_s \cdot \partial y_s/\partial j$ where dc/dy_s is calculated along the ENTER edge, whilst δc is given by $\partial c/\partial x_s \cdot \partial x_s/\partial i$. The first derivative in the latter product is computed for each face as follows:

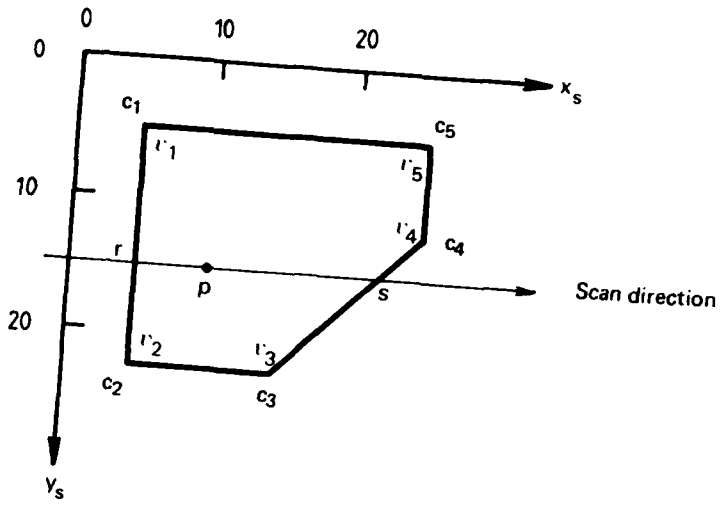
assume

$$c = kx_s + ly_s + m\,; \quad k, l, m \text{ constant}$$

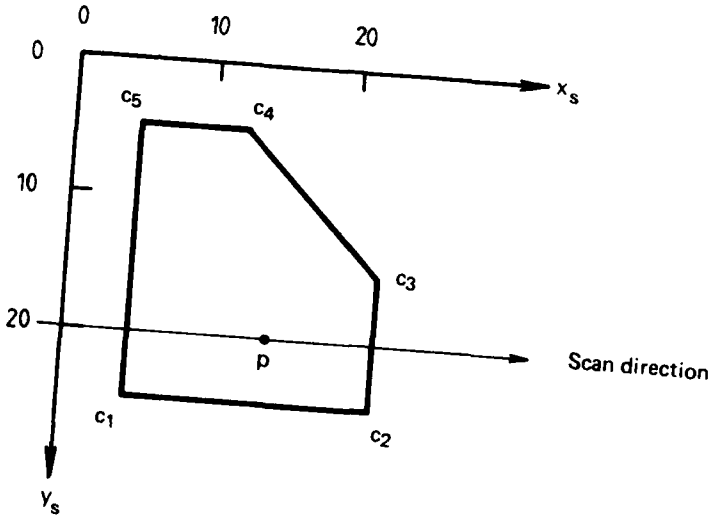
then

$$\frac{\partial c}{\partial x_s} = k$$

and, by virtue of the perspective transformation of Section 3.4 and Cramer's rule, any three non-collinear vertices of a face will give



(a)



(b)

FIG. 5.1 SHADING VARIATION WITH SCREEN LINE ORIENTATION.

$$\frac{\partial c}{\partial x_s} = \det \begin{pmatrix} c_1 x_1 & z_1 d + x_1 \bar{y} & x_1 \\ c_2 x_2 & z_2 d + x_2 \bar{y} & x_2 \\ c_3 x_3 & z_3 d + x_3 \bar{y} & x_3 \end{pmatrix} \Bigg/ \det \begin{pmatrix} y_1 d + x_1 \bar{x} & z_1 d + x_1 \bar{y} & x_1 \\ y_2 d + x_2 \bar{x} & z_2 d + x_2 \bar{y} & x_2 \\ y_3 d + x_3 \bar{x} & z_3 d + x_3 \bar{y} & x_3 \end{pmatrix}$$

Assuming $c(i_{min}, j_{min})$, Δc and δc appear in the description of each ENTER edge, realization of Gouraud's method is achieved by the following extensions to scan line algorithm;

LINE PHASE:

at each successive scan line the colour of the first pixel of the face encountered by the scan is given by

$$c(i_{k+1}, k+1) = c(i_k, k) + \Delta c ;$$

$k = j_{min}, \ldots$ and where i_k denotes the pixel corresponding to $j = k$ on the ENTER edge and $c(i_{k+1}, k+)$ replaces $c(i_k, k)$ in the edge description. This is analogous to the recursion in Section 4.2.

ELEMENT PHASE:

for each segment $[i_1, i_2]$ of the scan line spanned by the surface, the colour at pixel i_1 is computed by

$$c(i_1, k) = \delta c \ (i_k - i_1) + c(i_k, k)$$

This value together with the colour increment δc is passed to the relevant output field with i_1. The display hardware then computes the pixel colours on the span by the recursion

$$c(i+1, k) = c(i, k) + \delta c ; \quad i = i_1, \ldots, i_2.$$

It is worthwhile to note that the shading algorithm is useful in simulating the atmospheric fading of extended objects such as runways. In this case the colour of each vertex is computed as a function of surface colour and vertex–observer distance, and the colour of any surface point is determined by interpolation. As discussed in [1] (Section 6.4) the desaturation of the sky in the vicinity of the horizon is accomplished in a similar manner.

In contrast, atmospheric fading of objects of relatively small extent does not require the shading algorithm. Instead it is sufficient that the assigned colour of an object is computed as function of its intrinsic colour and distance from the observer [1], [4]. This computation is conviently executed prior to the scan line algorithm.

5.2 Edge Smoothing

The CGI process can be viewed as a two dimensional sampled data system, with each pixel being assigned the colour of the surface visible at the centre of the pixel. The attendant limited resolution is responsible for image anomalies referred to as "step effects". A typical example of this is the staircase appearance of near-vertical and near-horizontal edges.

A number of ways of eliminating these effects have been reported in the literature [4], [7]. In the present context the most appealing method is based on the "area-averaging" approach of [4], in which the colour assigned to a pixel bisected by an edge is the blend, based on subtended area, of the colours on either side of the edge. Specifically, referring to Figure 5.2, the edge bisects the pixel into two regions of area A and B, with respective colours c_A, c_B. Then the colour assigned to the pixel is simply $c = Ac_A + Bc_B$; assuming pixel is of unit area.

For efficient mechanization of this approach, near-vertical and near-horizontal edges have to be treated separately. In the case of near-vertical edges, the preceding computation suffices since at most only a few pixels per scan line will be involved. On the other hand for near-horizontal edges, the edge may bisect many pixels, thus requiring excessive computational effect to determine the subtended area A, B at each pixel. A more efficient approach is to employ linear variation of colour along the edge [4], analogously to the curved surface shading algorithm of the previous section. With reference to Figure 5.3, for a near-horizontal edge, its span across the scan line is determined, together with the colour on either side of the edge. If the edge spans n pixels (from i_1 to $i_1 + n - 1$) then the colour assigned to pixels bisected by the edge is given by the recursion

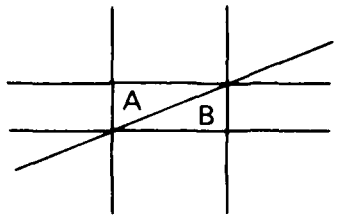


FIG. 5.2 EDGE BISECTING A PIXEL.

$$c(i+1,k) \quad c(i,k) + \frac{c_B - c_A}{n+1}; \quad c(i_1,k) \quad \frac{n}{n+1} c_A + \frac{1}{n+1} c_B$$

$$i = i_1, i_1 + 1, \ldots, i_1 + n - 1$$

Although the recursion is not strictly correct for small n, indications are that the attendant error is not visually significant ([4], p. 163).

Within the framework of the present scan line algorithm an additional buffer register, BR1, is required at the end of the Element Phase such that BR and BR1 refer to the surfaces (hence colours) visible on either side of each edge, whilst edge slope information is passed from the EDGE list. Referring to Figure 5.4, at each new visible surface determined by the Element Phase, the contents of BR are copied to BR1, and the new visible surface is referenced by BR. Thus BR1 refers to the colour to the left of the edge whilst BR refers to the colour to the right of the edge. The slope of the edge then determines whether the near-vertical or near-horizontal algorithm is to be invoked. In each case once the smoothing has been completed, control passes onto the shading algorithm.

6. CONCLUDING REMARKS

This report describes the realization of a scan line algorithm to solve the hidden surface problem within the framework of the total CGI process. The algorithm operates on a depth-sorted list of visible image polygons, and utilizes image coherence in conjunction with a priority list to compute the visible surface at each picture element.

The analysis of a software realization of the CGI process indicates that the scan line algorithm is the most significant functional block in terms of computational load. As for the scan line algorithm, the performance analysis demonstrates that the resolution of the visibility problem at each scan line edge intersect dominates the computation, and confirms the relative importance of sorting to the process. Finally in the context of manned flight simulation, the analysis implies that the achievement of real time CGI requires hardware realization of the scan line algorithm.

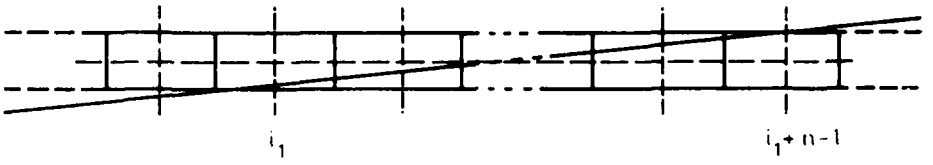


FIG. 5.3 SPAN OF A NEAR–HORIZONTAL EDGE.

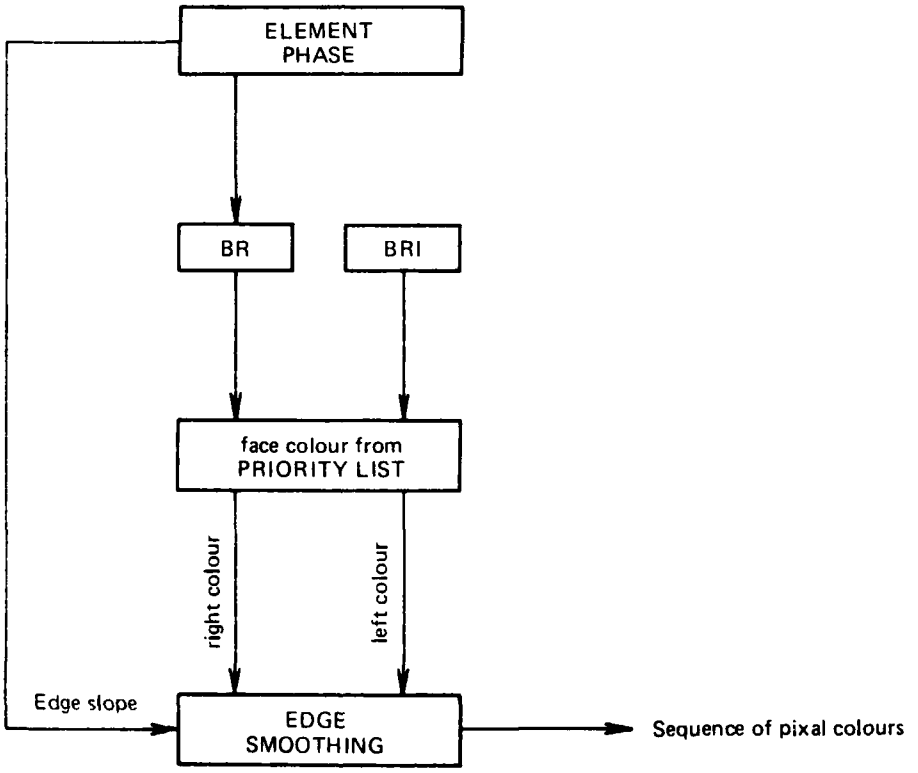


FIG. 5.4 INCORPORATION OF EDGE SMOOTHING IN CGI PROCESS STRUCTURE.

REFERENCES

[1] Sandor, J.—"Computer synthesis of flight simulation visuals" ARL-Sys-Note-61, Aeronautical Research Laboratories, Feb. 1979.

[2] Phong, B. T.—"Illumination for computer generated pictures", Comm. ACM, Vol. 18, No. 6, June 1975, pp. 623–629.

[3] Sutherland, I. E., and Hodgman, G. W.—"Re-entrant polygon clipping", Comm. ACM, Vol. 17, No. 1, Jun. 1974, pp. 32–42.

[4] Beardsley, H., *et al.*—"Advanced simulation in undergraduate pilot training", AFHRL-TR-75-59(V), U.S. Air Force Human Resources Laboratory, Nov. 1975.

[5] Gouraud, H.—"Continuous shading of curved surfaces", IEEE Trans. Computers, Vol. C-20, No. 6, June 1971, pp. 623–629.

[6] Sutherland, I. E. *et al.*—"A characterization of ten hidden-surface algorithm", Computing Surveys, Vol. 6, No. 1, March 1974.

[7] Crow, F. C.—"The aliasing problem in computer-generated shaded images", Comm. ACM, Vol. 20, No. 11, November 1977, pp. 799–805.

[8] Etkin, B.—"Dynamics of Atmospheric Flight". New York: Wiley, 1972.

APPENDIX 1

Transformations

Let a vertex v have coordinates (x_L, y_L, z_L) in the local coordinate frame $O_L X_L Y_L Z_L$ and let O_L be located at $(o_{L_x}^E, o_{L_y}^E, o_{L_z}^E)$ in $O_E X_E Y_E Z_E$. Referring to Figure A1 the coordinate frame $\hat{O}_E \hat{X}_E \hat{Y}_E \hat{Z}_E$, located at O_E, parallel to $O_L X_L Y_L Z_L$, is brought into coincidence with $O_E X_E Y_E Z_E$ by the sequence of rotations [8]

(i) rotation ϕ_1 about $O_E \hat{Z}_E$, carrying $O_E \hat{X}_E \hat{Y}_E \hat{Z}_E$ to $O_E X_2 Y_2 Z_2$

(ii) rotation ϕ_2 about $O_E Y_2$, carrying $O_E X_2 Y_2 Z_2$ to $O_E X_E Y_3 Z_3$

(iii) rotation ϕ_3 about $O_E X_E$, carrying $O_E X_E Y_3 Z_3$ to $O_E X_E Y_E Z_E$

such that v will have coordinates (x_E, y_E, z_E) in $O_E Y_E Y_E Z_E$ given by

$$\begin{pmatrix} x_E \\ y_E \\ z_E \end{pmatrix} = T(\phi_3, \phi_2, \phi_1) \begin{pmatrix} x_L + o_{L_x}^E \\ y_L + o_{L_y}^E \\ z_L + o_{L_x}^E \end{pmatrix} ; \quad T : R_3 \to R_3$$

where

$$T(\phi_3, \phi_2, \phi_1) = \begin{pmatrix} \cos\phi_1 \cos\phi_2 & \cos\phi_2 \sin\phi_1 & -\sin\phi_2 \\ \cos\phi_1 \sin\phi_2 \sin\phi_3 & \sin\phi_1 \sin\phi_2 \sin\phi_3 & \cos\phi_2 \sin\phi_3 \\ \quad - \sin\phi_1 \cos\phi_3 & \quad + \cos\phi_1 \cos\phi_3 & \\ \cos\phi_1 \sin\phi_2 \cos\phi_3 & \sin\phi_1 \sin\phi_2 \cos\phi_3 & \cos\phi_2 \cos\phi_3 \\ \quad + \sin\phi_1 \sin\phi_3 & \quad - \cos\phi_1 \sin\phi_3 & \end{pmatrix}$$

Similarly if O has coordinates (o_x^E, o_y^E, o_z^E) in $O_E X_E Y_E Z_E$ and $\hat{O}\hat{X}\hat{Y}\hat{Z}$ is the frame located at (o_x^E, o_y^E, o_z^E), parallel to $O_E X_E Y_E Z_E$ then v has coordinates (x, y, z) in $OXYZ$ given by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = T(\theta_3, \theta_2, \theta_1) \quad \begin{pmatrix} x_E - o_x^E \\ y_E - o_y^E \\ z_E - o_z^E \end{pmatrix}$$

where $\theta_1, \theta_2, \theta_3$ and $T(\theta_1, \theta_3, \theta_3)$ are analogous to ϕ_1, ϕ_2, ϕ_3 and $T(\phi_1, \phi_2, \phi_3)$ respectively.

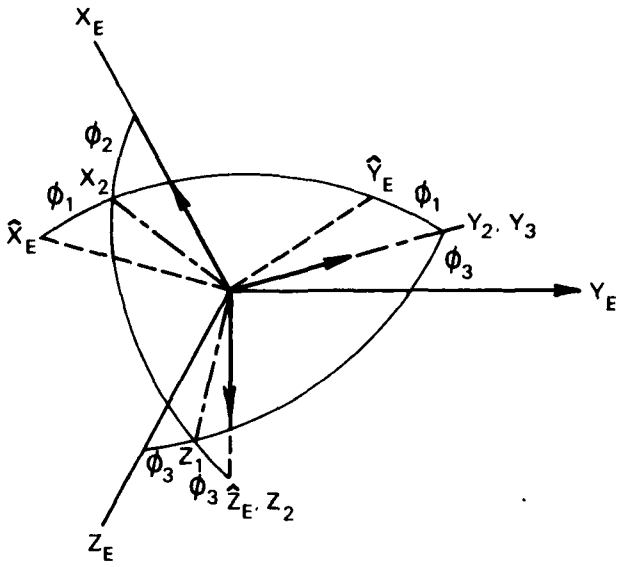


FIG. A1 SEQUENCE OF ROTATIONS ϕ_1, ϕ_2, ϕ_3

DISTRIBUTION

	Copy No.
AUSTRALIA	
Department of Defence	
Central Office	
Chief Defence Scientist	1
Deputy Chief Defence Scientist	2
Superintendent, Science and Technology Programmes	3
Australian Defence Scientific and Technical Representative (UK)	—
Counsellor, Defence Science (USA)	—
Joint Intelligence Organisation	4
Defence Central Library	5
Document Exchange Centre, D.I.S.B.	6–22
DGAD (NCO)	23
Aeronautical Research Laboratories	
Chief Superintendent	24
Library	25
Superintendent—Systems Division	26
Divisional File—Systems	27
Authors: J. Sandor	28
P. G. Jost	29
Materials Research Laboratories	
Library	30
Defence Research Centre, Salisbury	
Library	31
Central Studies Establishment	
Information Centre	32
RAN Research Laboratory	
Library	33
Navy Office	
Naval Scientific Adviser	34
Army Office	
Army Scientific Adviser	35
Air Force Office	
Aircraft Research and Development Unit, Scientific Flight Group	36
Air Force Scientific Adviser	37
Technical Division Library	38
DGAIRENG	39
RAAF Academy, Point Cook	40
Department of Industry and Commerce	
Government Aircraft Factories	
Manager	41
Library	42

SWITZERLAND

Institute of Aerodynamics, Library 72

UNITED KINGDOM

Aeronautical Research Council, Secretary 73
CAARC, Secretary 74
Royal Aircraft Establishment:
Farnborough, Library 75
Bedford, Library 76
British Library, Lending Division 77
Aircraft Research Association, Library 78
Science Museum Library 79
British Aerospace Corporation, Kingston-Brough, Library 80

Universities and Colleges

London	Professor A. D. Young, Aero. Engineering	81
Belfast	Dr A. Q. Chapleo, Dept. of Aeron. Eng.	82
Cranfield Inst. of Technology	Library	83
Imperial College	The Head	84

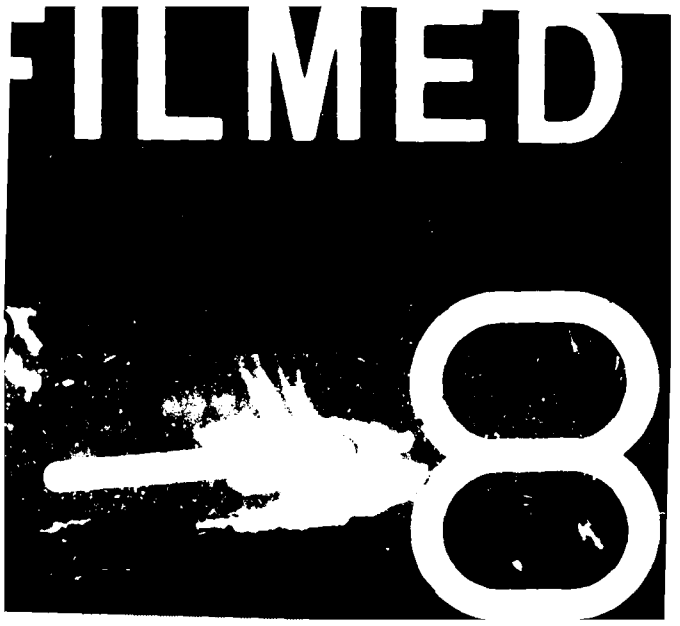
UNITED STATES OF AMERICA

NASA Scientific and Technical Information Facility 85
American Institute of Aeronautics and Astronautics 86
Bell Helicopter Textron 87
Boeing Co. Head Office, Mr R. Watson 88
Cessna Aircraft Co., Executive Engineer 89
Lockheed California Company 90
Calspan Corporation 91
Air Force Human Resources Lab., Advanced Systems Division 92
Naval Training and Equipment Centre 93
Evans and Sutherland, Advanced Visual Systems, Director 94
McDonnell *Douglas Electronics Company* 95
The Singer Company, Link Division 96

Universities and Colleges

Florida	Aero. Engineering Dept.	97
Stanford	Dept. of Aero. Library	98
Polytechnic Inst. of New York	Aeronautical Labs., Library	99
California Inst. of Technology	Graduate Aeronautical Labs., Library	100
Massachusetts Inst. of Technology	Library	101
New York Inst. of Technology	Computer Graphics Laboratory	102

Spares 103–112


FILMED
8